UNIVERSIDAD AUTÓNOMA DE MADRID

ESCUELA POLITÉCNICA SUPERIOR

TRABAJO FIN DE GRADO

EL PROCESO DE REQUISITOS EN EL DESARROLLO DE OPEN SOURCE SOFTWARE

Autora: Olympia Muruzábal Ishigetani
Tutora: Silvia Teresita Acuña Castillo

SEPTIEMBRE 2014

Agradecimientos

Todo este trabajo no habría sido posible sin el apoyo directo o indirecto de mucha gente, a las cuales me gustaría dedicar esta página para agradecérselo.

En primer lugar, dar las gracias a mi tutora Silvia, por su supervisión y guía en todo el proceso seguido en la investigación y por su especial atención en los detalles para la elaboración de este documento.

A mis profesores a lo largo de la carrera, que compartieron sus conocimientos conmigo, y a mis compañeros, que en estos años han conseguido hacer más ameno el estudio y las interminables horas en los laboratorios.

En especial agradecer a Cristina su paciencia conmigo como compañera de fatigas, por todos sus consejos, apoyo incondicional y por estar siempre a mi lado como una hermana.

Y por último pero no menos importante, a mi familia por motivarme y apoyarme en todo momento, por el sacrificio y la dedicación por darme la mejor educación posible tanto profesional como personalmente.

Muchas gracias.

Resumen

El estudio realizado en este trabajo fin de grado se enmarca en el campo de la Ingeniería del Software. Más concretamente, el proyecto se encuadra en el análisis de las actividades de requisitos y métodos de trabajo del proceso de desarrollo de *open source software* (OSS) y sus comunidades de usuarios y desarrolladores. Aunque actualmente el OSS está adquiriendo un gran éxito tanto a nivel de usuario como industrial, todavía se desconoce exactamente lo que hacen exitosos a los productos OSS y a sus procesos de desarrollo.

El propósito de este trabajo es determinar si las actividades que se llevan a cabo en el proceso de requisitos en comunidades OSS, a diferencia del proceso de requisitos de la Ingeniería del Software tradicional, es una de las claves del éxito del software de código abierto. El proceso de requisitos forma parte de la etapa inicial de un proyecto y es imprescindible puesto que define las funcionalidades del software a desarrollar y da soporte a la adaptación y evolución del proyecto. En concreto, los objetivos de este trabajo consisten, precisamente, en analizar y determinar cómo se llevan a cabo los procesos y actividades relacionadas con la Ingeniería de Requisitos en comunidades OSS y compararlas con las realizadas en el proceso de requisitos tradicional.

Después de describir el estado del OSS y realizar un estudio previo sobre las actividades de la Ingeniería de Requisitos en el software tradicional, se seleccionan cuatro comunidades OSS como casos de estudio: OpenOffice, Mozilla, NetBeans y Eclipse. Se trata de comunidades con miles de participantes, donde se procede a investigar y analizar de forma detallada las actividades de educción, análisis, especificación y gestión de requisitos. En otras comunidades más pequeñas: DoubleCommander, FileZilla y Social Network Manager, se estudia el proceso para realizar una aportación desde el punto de vista de un desarrollador.

No podemos generalizar los resultados puesto que los casos de estudio son reducidos. Aun así, en los resultados obtenidos podemos observar claras diferencias entre el proceso tradicional y el seguido en las comunidades OSS estudiadas. En concreto, el proceso tradicional se trata de un proceso formalizado y controlado con actividades bien definidas. Al contrario el OSS no sigue ningún estándar ni obligación, lo que se percibe en la falta de documentación formal. En las comunidades OSS los usuarios pueden

colaborar bajo la supervisión de los administradores o moderadores, dependiendo de la cantidad de colaboradores y del gobierno que rija la comunidad la toma de decisiones puede ser más estricta.

A través del estudio del proceso de requisitos seguido en los desarrollos OSS es posible comprender sus prácticas, en las que participan tanto usuarios como desarrolladores. En este trabajo se determina cómo estos se comunican y realizan la gestión de requisitos por medio de infraestructuras online como son los foros, listas de correo, grupos de noticias, wikis, y cómo se pueden realizar aportaciones por medio de herramientas accesibles desde la Web. Estas prácticas pueden ser transferidas al proceso de desarrollo tradicional, ya sea en el desarrollo del proyecto o en la fase de mantenimiento. Conocer el proceso de desarrollo OSS ayuda a que futuros voluntarios participen en los proyectos OSS al conocer cómo se desarrolla software en las comunidades, favoreciendo la creación y evolución de nuevos proyectos exitosos.

Palabras Clave: Open Source Software, Proceso de Requisitos, Ingeniería de Requisitos, Comunidad Open Source.

Abstract

The research and development reported in this final-year project addresses the field of software engineering. To be precise, the project analyses the requirements activities and working methods used in the open source software (OSS) development process and by OSS user and developer communities. Although OSS is now becoming very popular among users and in industry, it is still unclear what really makes OSS development processes and their products successful.

The purpose of this study is to determine whether, compared with the traditional software engineering requirements process, the activities performed as part of the requirements process in OSS communities are one of the OSS success factors. The requirements process is one of the early stages of a project and is essential since it defines the functionality of the software to be developed and supports project adaptation and evolution. Specifically, the objectives of this project are precisely to analyse and determine how the processes and activities related to requirements engineering are enacted in OSS communities compared with traditional requirements process activities.

After describing the state of the art in OSS and conducting a preliminary study of traditional software requirements engineering activities, we selected several OSS communities as case studies. We thoroughly investigated and analysed the requirements activities of elicitation, analysis, specification and management in OpenOffice, Mozilla, NetBeans and Eclipse, which are communities with thousands of participants. In smaller communities, like DoubleCommander, FileZilla and Social Network Manager, we examined the contribution process from the point of view of a developer.

The results cannot be generalized because the case studies are limited. Even so, the results clearly highlight the differences between the traditional process and the processes enacted in the analysed OSS communities. Specifically, the traditional process is a specified and controlled process with well-defined activities, whereas OSS does not adhere to any standard or rule as indicated by the absence of formal documentation. In OSS communities, users sometimes collaborate under the supervision of administrators or moderators, and decision-making may be stricter depending on the number of members and the type of government.

By studying the requirements process enacted in OSS development it is possible to understand OSS practices in which both users and developers participate. This project determines how requirements can be communicated and managed through online infrastructures such as forums, mailing lists, newsgroups, wikis, and how contributions are made through tools accessible from the Web. These practices are transferable to the traditional development process in either the development or the maintenance phase. Familiarity with the OSS development process is useful for future OSS project volunteers as they learn how communities develop software. This is beneficial for the creation and evolution of successful new projects.

Keywords: Open Source Software, Requirements Process, Requirements Engineering, Open Source Community.

Índice

Índice de Tablas

Índice de Figuras

Capítulo 1
INTRODUCCIÓN

La utilización y la creciente influencia del *open source software* (OSS) en la industria del software ha generado oportunidades, retos y un gran interés desde diferentes perspectivas, que van desde estudios sociológicos para conocer la motivación de sus participantes hasta estudios tecnológicos para comprender los procesos de desarrollo e innovación [Scacchi et al., 2005].

El fenómeno OSS es el paradigma de desarrollo de software de manera colaborativa y abierta que ha dado como resultado no solo software de reconocida calidad, sino también un nuevo modo de desarrollo de software y nuevas estrategias empresariales o modelos de negocio. Se puede observar que empresas tales como Sun Microsystems, Oracle, IBM y Novell se benefician del uso de OSS [Hauge et al., 2007].

El desarrollo OSS se realiza gracias a la colaboración por parte de los usuarios y desarrolladores de las comunidades [Mockus et al., 2000][Mockus et al., 2002]. Donde es habitual que los propios usuarios participen en las decisiones de cómo construir el software que satisfaga sus necesidades y los desarrolladores participen en el diseño de la solución, la implementación del código y el mantenimiento del sistema. En particular, en los proyectos de desarrollo OSS los procesos que realizan las comunidades relacionados con el análisis de requisitos son diferentes a los de la Ingeniería de Requisitos tradicional [Acuña et al., 2012][Castro y Acuña, 2012], los cuales se caracterizan por ejemplo por la falta de especificaciones de requisitos de software explícitas.

La evolución de las formas y procesos de desarrollo que utilizan las comunidades OSS, junto con los grandes beneficios de dinero que obtienen con el desarrollo de sus aplicaciones y el éxito que están obteniendo en el mundo de la industria, hace más interesante el estudio sobre cuáles son las diferencias y similitudes de dichos procesos

utilizados por las comunidades OSS de los tradicionalmente aplicados en la Ingeniería de Requisitos. Más aún, gracias al acceso al código abierto se promociona la competencia en el mercado del software, además de contribuir al avance en la industria como sucede por ejemplo con Mozilla Firefox o Android. Aunque las aplicaciones OSS están adquiriendo un gran éxito tanto a nivel de usuario como industrial, todavía se desconoce exactamente lo que hacen exitosos a los productos OSS y a sus procesos de desarrollo.

El proceso de requisitos forma parte de la etapa inicial de un proyecto y es imprescindible puesto que define las funcionalidades del software a desarrollar y da soporte a la adaptación y evolución del proyecto. El proceso de requisitos constituye la base de todo el trabajo de ingeniería del desarrollo del software posterior. En el caso de la Ingeniería de Requisitos tradicional se compone de varias actividades: educción, análisis y negociación, especificación, validación y gestión de requisitos. Si no se realiza adecuadamente puede causar problemas, donde el coste para solucionarlos aumenta conforme se avanza en el proyecto. En la literatura no hay muchos estudios detallados sobre el proceso de requisitos que siguen las comunidades OSS, lo que nos conduce a realizar este trabajo para profundizar en la organización y el proceso de requisitos llevado a cabo en las comunidades OSS.

En concreto, actualmente existen pocos estudios dedicados al análisis de la forma en que las comunidades de desarrollo OSS realizan y caracterizan los requisitos que se generan al realizar un proyecto y el éxito que conlleva esta forma de definir los requisitos. El objetivo de este trabajo consiste, precisamente, en comprender y determinar cómo se llevan a cabo los procesos y actividades relacionadas con la Ingeniería de Requisitos en un conjunto de comunidades OSS.

Para poder alcanzar dicho objetivo, se seleccionarán varios proyectos OSS sobre los cuales se realizará el estudio del proceso de requisitos, teniendo como guía la descripción de las actividades relacionadas con la Ingeniería de Requisitos de la comunidad OSS y la tradicional [SWEBOK, 2004]. Sobre otras comunidades OSS se realizarán pequeñas aportaciones. Todo el proceso seguido para conocer el proceso de requisitos en las comunidades OSS seleccionadas será documentado en bitácoras, que se recogen en los anexos. Posteriormente, se identificarán los procesos e infraestructuras utilizadas en la comunidad OSS, de tal forma que puedan ser transferidas las prácticas potenciales de éxito a la academia y a la industria.

Este trabajo está dirigido a usuarios que quieran colaborar en las aplicaciones OSS que utilizan habitualmente y a desarrolladores o ingenieros de software que deseen comprender o participar en el proceso de requisitos en una comunidad OSS. En el caso de que un usuario encuentre un error o no esté satisfecho con las funcionalidades del software, podrá conocer qué proceso seguir para hacer consultas, realizar una propuesta de requisito y hacer un seguimiento del mismo hasta su implementación. Por otro lado, los desarrolladores podrán informarse sobre cómo incorporarse a la comunidad, cómo realizar aportes, y conocer las infraestructuras que se utilizan. Es decir, se dan las guías para aportar un nuevo requisito o si ya ha sido considerado por otra persona, cómo buscar o hacer el seguimiento, y de qué modo se puede interactuar con la comunidad.

Tras definir en este capítulo el alcance y el contexto del problema para el desarrollo del estudio, así como también el objetivo y la importancia del trabajo, en el siguiente Capítulo 2 se presenta el estado de la cuestión donde se analizan los trabajos relacionados más destacados. En el Capítulo 3 se detalla el método utilizado para la investigación. En el Capítulo 4 se describen las diferencias y similitudes del proceso de requisitos de *open source* y del desarrollo tradicional determinadas a partir del análisis de la literatura. El Capítulo 5 recoge los datos obtenidos de los casos de estudio y el análisis y síntesis de los resultados. En el Capítulo 6 se detallan las aportaciones realizadas a las comunidades. En el Capítulo 7 se muestran las conclusiones alcanzadas en el proyecto y el trabajo previsto para el futuro. Por último, podemos encontrar las referencias y los anexos, donde se recogen las bitácoras de la investigación, los ejemplos de mensajes de las comunidades OSS relevantes para este trabajo, una encuesta realizada a usuarios y las descripciones de herramientas utilizadas en OSS.

Capítulo 2

ESTADO DE LA CUESTIÓN

En este capítulo se presentan el estado de la cuestión y los temas relacionados con el desarrollo del trabajo. Se definen los conceptos básicos necesarios para comprender los procesos de la Ingeniería de Requisitos en los proyectos de las comunidades de desarrollo OSS. Comenzando por la definición de OSS, las licencias, las formas de trabajo y los roles de OSS, y los conceptos de la Ingeniería de Requisitos tradicional según el SWEBOK (del inglés, *Software Engineering Body of Knowledge*). A continuación, se describirán de manera genérica los procesos de requisitos llevados a cabo por las comunidades OSS, así como las métricas de gobernabilidad y sus infraestructuras, que posteriormente nos facilitarán el estudio de las comunidades OSS. Por último, se detallan las estrategias y métodos de investigación de la Ingeniería del Software Empírica, área en la que se enmarca este trabajo.

2.1 Open Source Software

Comenzaremos explicando en qué consiste una comunidad OSS, puesto que es un concepto al cual nos referiremos durante todo el trabajo.

2.1.1 Definición

Según [Gacek y Arief, 2004] el término OSS se refiere a un proceso de desarrollo de software que se basa en las colaboraciones de los desarrolladores a través de Internet desde diversos puntos geográficos. Sin embargo, existen distintas perspectivas sobre si el desarrollo en las comunidades OSS se diferencia del desarrollo del software tradicional.

Pese a los distintos puntos de vista sobre su desarrollo podemos encontrar ciertas características del OSS catalogadas por la *Free Software Foundation*. Estas características son: permitir ejecutar el programa con cualquier propósito, estudiar las funcionalidades del programa y adaptarlo a las necesidades de cada usuario, acceder al código fuente, distribuir copias, mejorar el programa y publicar las mejoras, para el beneficio de la comunidad OSS.

Siguiendo estos criterios se puede considerar que una aplicación es OSS si cumple estos 10 requisitos [Heliö, 2004]:

1. El software ha de poder ser distribuido libremente.
2. El código fuente debe estar disponible a todos los usuarios.
3. El software debe poseer derechos para realizar modificaciones o trabajos derivados.
4. Integridad del código fuente del autor.
5. Distribución de la licencia.
6. La licencia no debe ser específica de un producto.
7. La licencia no debe restringir otro software.
8. La licencia debe ser tecnológicamente neutral.
9. No existe discriminación a personas o grupos.
10. No hay discriminación contra áreas de iniciativa.

2.1.2 Licencias

El uso de las licencias adquiere cierto relieve en el éxito del OSS. Una licencia es un acuerdo o contrato entre un usuario y un desarrollador en el cual se describen las normas de adquisición y utilización del software [Gerea, 2007].

Las licencias establecen una serie de condiciones que garantizan algunas libertades básicas de redistribución, modificación y uso para los usuarios, aseguran algunas condiciones impuestas por los autores y garantizan que las publicaciones de componentes sean también software libre.

Las principales licencias conocidas son la *GNU Public License* (GPL), *Lesser GPL* (LGPL), *Artistic License* y la *Berkeley System Distribution* (BSD), además de la *Mozilla Public License* (MPL).

La mayor parte del software se ha distribuido con la licencia GPL, que garantiza la libertad completa. La LGPL surgió para utilizarse con las librerías de software y relacionarlas con el software de código propietario, alternativamente apareció la *Open Software License* (OSL). La función de anunciar los derechos de autor se recogen en la licencia BSD. La licencia MPL trata la conversión de un producto de software comercial a código abierto.

Existen organizaciones que se encargan de definir las características de las licencias, como pueden ser Proyecto Debian (DFSG), *Open Source Initiative* (OSI) o *Free Software Foundation* (FSF).

2.1.3 Formas de Trabajo de las Comunidades OSS

Es interesante conocer las características generales y estilos de desarrollo de las comunidades OSS para alcanzar el objetivo planteado.

No existe un modelo a escala mundial que defina cómo debe ser el desarrollo del software, aun así se pueden observar características comunes en las comunidades. Habitualmente los participantes utilizan seudónimos para ser identificados, los desarrolladores utilizan su tiempo libre para desarrollar el código fuente y realizar aportaciones que implican el reconocimiento del resto de los participantes, y los usuarios utilizan los tablones online, foros de discusión, listas de correo electrónico, chats y wikis para observar, participar y contribuir en el desarrollo del proyecto [Escribano, 2011].

Los proyectos OSS se basan en una estructura jerárquica, donde normalmente hay un responsable, elegido por la comunidad, que toma las decisiones. Al contrario que en el software propietario los usuarios realizan una gran actividad en la fase de desarrollo, aunque pueden surgir problemas como *cross-participations*; es decir, un usuario que participa en discusiones del mismo tema en paralelo en diferentes listas de correo.

Se pueden encontrar diferentes estilos de desarrollo en las comunidades OSS:

- **Compañía**: en este caso, la compañía elige a los desarrolladores que se encargarán de la realización del producto y al jefe de proyecto, el cual puede ser un grupo de personas.

- **Fundación**: es una organización sin fines de lucro, que se crea para financiar un proyecto grande cuando una empresa no puede hacerse cargo a causa de falta de recursos económicos para financiar el proyecto por sí solo.
- **Comité**: se crea cuando un proyecto es demasiado grande con el fin de proporcionar un foro para la toma de decisiones, las cuales no se deciden solamente por el responsable del proyecto.
- **Individuos**: este caso se da cuando en un proyecto se implican menos de cinco desarrolladores.
- **Linux kernel**: se basa en una jerarquía de confianza dentro de la comunidad de desarrollo, que ha evolucionado desde hace decenas de años.

2.1.4 Roles OSS

Para comprender mejor el funcionamiento de las comunidades OSS es interesante conocer los distintos roles que participan en el desarrollo de los proyectos. Los principales roles en la comunidad OSS son: propietario de un OSS, desarrollador principal, desarrolladores, reportero de problemas, validadores del sistema, soporte a usuarios y usuarios.

En el caso de la industria según [Hauge et al., 2007] tenemos los siguientes roles:

- **Proveedor de un OSS**: es la organización que tiene el control del proyecto, y consigue la comercialización y la publicidad de su producto.
- **Integradores de OSS**: la organización que utiliza componentes de OSS y realiza listas de requisitos.
- **Participantes en comunidades OSS**: utilizan el software y proporcionan soluciones a errores además de proponer requisitos.
- **Participantes internos en comunidades OSS**: son las organizaciones que colaboran o adoptan las técnicas de desarrollo de una comunidad OSS.

2.2 La Disciplina de la Ingeniería de Requisitos

Para poder realizar una comparativa entre el proceso de requisitos en OSS y la Ingeniería de Requisitos tradicional, es necesario conocer sus características. En este

apartado se definirán los conceptos básicos de la Ingeniería de Requisitos, detallados en [SWEBOK, 2004].

2.2.1 Fundamentos de los Requisitos del Software

El área de conocimiento de la Ingeniería de Requisitos de la disciplina de la Ingeniería del Software se refiere a la educción, el análisis, la especificación, la validación y la gestión de los requisitos del software. Los requisitos del software expresan las necesidades y las expectativas en el producto de software que contribuye a la solución de un cierto problema del mundo real. Una característica imprescindible de los requisitos es que sean comprobables, de modo que posteriormente se puedan verificar. Además se les puede atribuir una prioridad, facilitando así un mejor control sobre los mismos [Kotonya y Sommerville, 2000].

Los requisitos se pueden clasificar de distintas formas, entre ellos como funcionales o no funcionales. Los requisitos funcionales describen las operaciones que el software va a ejecutar. Los requisitos no funcionales definen las cualidades o propiedades que debería tener el software. Pueden clasificarse en distintos tipos, como requisitos de usabilidad, fiabilidad, rendimiento, disponibilidad, mantenibilidad, seguridad, etc.

Los parámetros del producto son requisitos de software que hay que desarrollar, mientras que los parámetros del proceso son condiciones en el desarrollo del software. Las propiedades emergentes son características del software que dependen de otros componentes y de la arquitectura del sistema. La definición de sistema dada por [INCOSE, 2000] es: una combinación recíproca de los elementos para lograr un objetivo definido. Incluyendo el hardware, software, soporte lógico inalterable, gente, información, técnicas, instalaciones, servicios, y otros elementos de apoyo.

Los requisitos del software han de definirse lo más clara e inequívocamente posible evitando requisitos imprecisos, es decir que no dependan de una interpretación subjetiva para realizar la verificación. No se trata solo de escribir lo que el cliente quiere, los requisitos son dinámicos tanto como la comprensión del problema y es importante que sean precisos, correctos y flexibles [Bray, 2002].

2.2.2 Proceso de Requisitos de Software

El proceso de requisitos se inicia al principio del proyecto y se va refinando conforme avanza el ciclo de vida del software [SWEBOK, 2004]. Se identifican y se gestionan los requisitos, además de realizar las actividades de educción, análisis, especificación y validación.

Existen varios roles que participan en el proceso de requisitos, desde los usuarios y clientes y *stakeholders* (interesados) que utilizan el software, hasta los ingenieros de software, analistas de mercado y reguladores que comprueban el cumplimiento de las normativas.

En el proceso de requisitos se indican los recursos requeridos y consumidos en el grupo de actividades de requisitos, el coste, los recursos humanos, las pruebas y las herramientas. Se definen los estándares de calidad y los modelos de mejora de procesos software, las métricas en el proceso de requisitos, hitos y planes de mejora y puesta en práctica. El ciclo de vida de la Ingeniería de Requisitos consta de las siguientes actividades [SWEBOK, 2004]:

- **Educción**: identifica a los *stakeholders* que proporcionan los objetivos, necesidades, expectativas y límites del sistema. Para lo que existen distintas técnicas de obtención como son las entrevistas o encuestas, torbellino de ideas (*brainstorming*), casos de uso, prototipos, reuniones o la observación. Es importante destacar la comunicación entre los interesados (*stakeholders*) y los ingenieros de software, dado que la captura de requisitos es uno de los pasos cruciales en el proceso de requisitos.
- **Análisis**: estudia los requisitos obtenidos y comprueba su consistencia, completitud y correctitud. Los modelos facilitan el desarrollo de la solución de un problema del mundo real. Existen distintos modelos como los diagramas de control de flujo, modelos de estados, objetos o datos, interacciones entre usuarios, entre otros. Lo que determina la elección de un modelo u otro son las características del proyecto como la naturaleza del problema, la experiencia del ingeniero de software, los requisitos del cliente y la disponibilidad de métodos y herramientas.
- **Negociación**: estima las necesidades que pueden entrar en conflicto. Los problemas pueden surgir como resultado del análisis, ya sea porque varios

stakeholders incluyan requisitos incompatibles, de modo que es necesario consultar con los *stakeholders* para alcanzar un consenso.

- **Especificación**: se refiere a la generación de documentos de los requisitos de manera que los *stakeholders* y desarrolladores puedan entenderse. Dependiendo de la complejidad del sistema se elaboran tres tipos de documentos: a) documento de definición del sistema, enumera los requisitos del sistema desde la perspectiva del dominio; b) especificación de requisitos del sistema, donde se especifican los requisitos del sistema y los requisitos del software que se derivan de ellos; y c) especificación de requisitos del software, este documento establece la base para un acuerdo entre los clientes y los proveedores tanto sobre lo que tiene que hacer como lo que no se espera del producto software, proporciona también una estimación de los costes, los riesgos y la planificación. Los requisitos del software se escriben en lenguaje semi-formal, lo que permite desarrollar la validación y la verificación, además de informar con mayor facilidad a los usuarios.
- **Validación**: es el proceso de examinar el documento de requisitos para asegurarse de que éste define el software correctamente y que el ingeniero de software comprende los requisitos. El medio más común de realizar la validación es la inspección o revisión, que se realiza por un grupo compuesto al menos de un representante del cliente. Los prototipos son comúnmente el medio utilizado para validar la interpretación del ingeniero de software sobre los requisitos, así como para definir nuevos requisitos. Una tarea importante es planear cómo verificar cada requisito, identificando y diseñando pruebas de aceptación que permitirán una validación del producto final.
- **Gestión**: mantiene un control sobre los cambios que se puedan producir durante el desarrollo de la aplicación, es decir, define los procedimientos necesarios y realiza el análisis que se debe aplicar a los cambios propuestos.

Los requisitos iteran típicamente hacia un nivel de calidad y detalle mayor que permiten las decisiones de diseño. El entendimiento de los requisitos continúa desarrollándose en el diseño y el desarrollo. Por lo que, el proceso de requisitos no es simplemente una tarea anticipada en el desarrollo del software, sino su alcance es continuo y abarca todo el ciclo de vida del software.

2.3 Gestión de Requisitos en OSS

En este apartado nos centraremos en la gestión de los requisitos para el desarrollo del software en las comunidades OSS. Los requisitos de proyectos OSS se recopilan en Internet, no se especifican de manera formal en documentos, como es el caso de organizaciones de software propietario. Se pueden encontrar infraestructuras que ofrecen servicios como las listas de correo electrónico, foros, chats o wikis. Una razón por la que no se documentan los requisitos es la gran familiarización de la comunidad con los requisitos del sistema.

Los procesos de la Ingeniería de Requisitos se realizan cara a cara, al contrario que en las comunidades de OSS que se realizan en línea a través de Internet. Los requisitos podemos encontrarlos organizados y guardados de forma persistente en infraestructuras de la comunidad, accesibles a nivel mundial.

2.3.1 Ingeniería de Requisitos en la Wiki

Las wikis proporcionan una plataforma flexible para la colaboración asíncrona a fin de crear contenido general que beneficia la participación. La fácil vinculación de una página reduce la redundancia y el historial de una página proporciona una base para la trazabilidad de los requisitos.

Para que todo funcione correctamente y con fluidez es importante tener en cuenta la estructura general de los temas de la wiki y proporcionar plantillas para la contribución de contenido. También es necesario que todos los usuarios conozcan las normas de uso, para que los nuevos interesados se incorporen más rápidamente al proyecto. Otro aspecto a tener en cuenta es que las comunicaciones y discusiones en la wiki han de ser moderadas y analizadas para gestionar conflictos y malentendidos.

Ejemplos sobre los documentos para la educción de requisitos pueden ser los siguientes: la página principal, información sobre las personas involucradas en el proyecto (*user homepage*), especificación breve de las interacciones del sistema (*user story*), representación estructurada del *user story* (*use case*) y los roles que participan en el *use case* y en el *user story* (actor).

2.3.2 Ingeniería de Requisitos en los Foros

Dado que el número de personas implicadas en el proceso de obtención de requisitos puede llegar a ser muy numeroso y difícil de coordinar, debido también a la dispersión geográfica, se depende de herramientas colaborativas. Las técnicas utilizadas por los usuarios de estas herramientas requieren de una colaboración y comunicación activa. Por ejemplo, es necesaria una forma de priorizar los requisitos según sus necesidades como obligatoria, deseada o no esencial o utilizando árboles binarios de búsqueda.

Un usuario puede realizar nuevas peticiones o búsquedas en el foro y los administradores se dedican a determinar la prioridad y la viabilidad de un requisito y de comunicar y actualizar el estado de los mismos. Se conocen tres formas diferentes para asignar las prioridades: a) un botón de votación; b) permitir a los *stakeholders* asignar la prioridad en una discusión; o c) los usuarios del foro expresan sus prioridades como parte de sus comentarios.

Un estudio de [Laurent y Cleland-Huang, 2009] demuestra que la mayoría de usuarios no realiza una búsqueda y que introduce directamente una nueva propuesta en un nuevo tema, lo que puede generar problemas de requisitos duplicados.

2.4 Gobernabilidad, Toma de Decisiones y Relaciones de Poder

Para entender mejor cómo funcionan las comunidades OSS y cómo se realizan las decisiones es necesario conocer los distintos tipos de liderazgo que podemos encontrar. Según Capra y colaboradores [Capra et al., 2008] hay cuatro aspectos que afectan a la gobernabilidad:

- El **código**, es 100% libre, salvo en proyectos comerciales OSS donde parte del código es cerrado y comercializado con licencias.
- La **contribución**, indica la cantidad de código desarrollado libremente.
- El **liderazgo**, consiste en la jerarquía de la estructura de coordinación de un proyecto.
- Las **prácticas de trabajo**, se trata de las prácticas de comunicación de un proyecto que se encuentra distribuido geográficamente y de forma virtual.

En las Tablas 1 y 2 se definen las siguientes dimensiones de liderazgo y prácticas de trabajo [Capra et al., 2008]:

Valor	Liderazgo	Descripción
1	**Organización predominante**	El proceso de desarrollo es liderado por una organización (compañía, institución o comité) la cual tiene un rol de liderazgo predominante, toma las decisiones y define un calendario formal para el proyecto.
2	**Organización o dictador benevolente**	El proceso de desarrollo es liderado por una organización (compañía, institución o comité) o por un dictador benevolente que define un calendario formal para el proyecto. La comunidad está fuertemente involucrada en el proceso de toma de decisiones, aunque finalmente las decisiones son tomadas por el líder del proyecto.
3	**Comunidad con principios comunes y reglas formales**	La comunidad se rige por principios comunes y reglas formales. Las decisiones se toman principalmente a través de sistemas de votación u órganos de gobierno directamente escogidos por los contribuyentes.
4	**Comunidad sin organización formal**	La comunidad carece de una organización formal y un órgano de gobierno, donde las decisiones son tomadas a través de discusiones informales.
5	**Líder + Organización formal**	La comunidad tiene un líder y una organización formal, el cual está involucrado en la fase de la toma de decisiones del proyecto. Sin embargo, los desarrolladores, contribuidores y usuarios finales están fuertemente implicados en la toma de decisiones a través de sistemas de votación junto con discusiones informales realizadas a través de infraestructuras online.

Tabla 1. Dimensiones de liderazgo de un gobierno

Valor	Prácticas	Descripción
1	**In situ**	Los desarrolladores trabajan en el mismo lugar, se comunican cara a cara y realizan reuniones físicas regulares.
2	**In situ + Infraestructuras online**	La mayoría de desarrolladores trabajan en el mismo lugar y realizan reuniones físicas. Los equipos pueden utilizar herramientas de comunicación virtual.
3	**Infraestructuras online + Reuniones físicas**	La comunidad está dispersa y la mayoría de desarrolladores se comunican a través de herramientas de comunicación virtual. Un subconjunto de desarrolladores puede trabajar en el mismo lugar y reunirse regularmente.
4	**Infraestructuras online**	La comunidad está dispersa y todos los desarrolladores se comunican de forma online. Carecen totalmente de reuniones físicas o raramente se llevan a cabo.

Tabla 2. Dimensiones de las prácticas de trabajo

A diferencia del software privado donde el cliente define los requisitos necesarios, en las comunidades OSS los usuarios participan en todas las actividades de desarrollo de un proyecto y es el mismo desarrollador el que define los requisitos.

Habitualmente en el desarrollo de software propietario existe una jerarquía formada por un jefe de proyecto y los desarrolladores, pero en las comunidades OSS no se define como tal. Dependiendo del tamaño del proyecto pueden surgir responsables de determinadas tareas. En el caso de que haya una colaboración con una empresa, ésta debe adaptarse a la comunidad.

La experiencia y el cargo de poder se obtiene principalmente por la educación adquirida y la experiencia laboral, en el caso del software privado. Mientras que en la comunidad OSS se obtiene mediante las aportaciones de calidad.

2.5 Principales Infraestructuras

Hoy en día podemos encontrar muchas infraestructuras donde localizar las comunidades OSS, entre ellas cabe destacar las siguientes:

Ohloh.net, se trata de un directorio público y gratuito de personas y OSS, donde se proporcionan distintos servicios como el seguimiento de errores o Google Analytics, pero no se alojan los proyectos OSS [Ohloh, 2014].

SourceForge.net, actualmente es el sitio web más grande de desarrollo OSS. Su éxito se debe a los servicios gratuitos que proporciona: *hosting* de código y web, un sistema

de gestión de errores, foros, listas de correos, wikis o blogs, la distribución de archivos, estadísticas y una amplia comunidad [SourceForge, 2014].

2.6 Ingeniería del Software Empírica

En este apartado describiremos las estrategias y métodos empíricos que se utilizan en la Ingeniería del Software y que posteriormente aplicaremos en este trabajo, los detalles de la elección y el procedimiento seguido para su ejecución se describen en el Capítulo 3, Método de Investigación.

Las investigaciones se dividen en dos tipos de paradigmas: cuantitativas, que recogen y analizan datos cuantitativos, o cualitativas, en las que se realizan registros narrativos de los casos estudiados. La diferencia es que la investigación cuantitativa estudia la relación entre datos cuantificados, mientras que la investigación cualitativa lo hace en contextos estructurales y situacionales.

[Runesson y Höst, 2009] definen cuatro tipos de estrategias según el propósito de la investigación:

- **Exploratoria**: trata de descubrir lo que está sucediendo, buscar nuevas ideas o generar ideas e hipótesis para nuevas investigaciones. Es un estudio previo a una investigación mayor.
- **Descriptiva**: retrata una situación o fenómeno, reafirmando características o atributos de una población.
- **Explicativa**: busca la explicación a una situación o a un problema, sobre todo pero no necesariamente en forma de una relación causal.
- **Mejora**: trata de mejorar un determinado aspecto de un fenómeno estudiado.

Hay tres métodos de investigación en la Ingeniería del Software [Robson, 1994]:

- **Estudio de casos**: consiste en la investigación de un solo fenómeno durante un tiempo específico, de modo que es difícil generalizar e interpretar los resultados. Para realizar un seguimiento adecuado se siguen una serie de directrices: definir las hipótesis o preguntas de investigación, seleccionar los proyectos piloto, identificar el método de comparación, minimizar el efecto de factores de confusión, planear los casos de estudio, supervisar el caso de estudio en relación con el plan y analizar e informar de los resultados.

- **Experimento formal**: se realiza en un entorno de laboratorio donde es posible controlar las distintas variables. El problema que se encuentra es el grado de control sobre las variables y la escala del problema, que normalmente es reducida.
- ***Survey***: se trata una investigación retrospectiva donde se recoge información de una población específica o de un individuo concreto, habitualmente se realizan por medio de cuestionarios o entrevistas.

Los experimentos formales son cuantitativos, mientras que los casos de estudio y los *surveys* pueden ser cuantitativos o cualitativos.

Capítulo 3

MÉTODO DE INVESTIGACIÓN

En este capítulo se detallan la estrategia y el método empírico utilizados para la presente investigación. Dado que no existen muchos estudios ni información sobre los procesos de requisitos utilizados por las comunidades OSS, lo más conveniente es realizar una estrategia empírica de tipo exploratoria. De este modo, respondiendo a una serie de preguntas podemos obtener información y datos que nos permitirán realizar un estudio posterior más profundo.

Tal y como se describe en el apartado 2.6 del Capítulo 2, Estado de la Cuestión, la estrategia exploratoria suele ser un estudio donde la información se recopila y analiza como un paso previo a un estudio de mayor envergadura. En general, lo que ofrece el estudio exploratorio son nuevos datos que pueden ser analizados con mayor profundidad o de forma más específica en un estudio posterior. En nuestro caso aprovecharemos estos datos para verificar las afirmaciones planteadas en el Capítulo 4, que caracterizarían el proceso de requisitos de OSS.

Una vez se han conocido y comprobado las características del proceso de requisitos de OSS, se seleccionan proyectos pequeños de OSS y se vincula a su comunidad de desarrollo, realizando un conjunto de aportes o entrevistando a un desarrollador principal para identificar diferencias y similitudes entre los procesos de requisitos de comunidades grandes y de proyectos pequeños.

Para ello utilizaremos el método de investigación llamado estudio de casos en el que cada caso será un proyecto OSS en el que nos adentraremos. El estudio de casos es un método de investigación que se basa en el principio de explorar algo a través de la propia experiencia del investigador (investigación empírica), mediante la observación y participación sin alteración del sujeto u objeto de estudio [Runesson y Höst, 2009]. La información que se trata en este tipo de estudio se basa en las propias experiencias del

investigador. Todas las actividades de cada caso serán almacenadas en bitácoras (documento en el que se anota todo lo realizado en relación con la comunidad del proyecto considerado, ya sean mensajes intercambiados, actividades realizadas, como opiniones personales).

Escogimos realizar un proceso iterativo enfocado en la investigación empírica sobre un conjunto de casos de estudio. Esto nos proporciona un diseño flexible para realizar modificaciones, como añadir más datos relevantes que no se consideraron en las primeras iteraciones. Los pasos del proceso seguido son los siguientes:

Paso 1. Elaboración del estado de la cuestión. El primer paso consiste en un estudio de la literatura existente, que se puede encontrar en los Capítulos 2 y 4.

Paso 2. Definición de la estrategia y del método. Teniendo en cuenta el objetivo y los recursos disponibles se ha adoptado una estrategia exploratoria y el método de estudio de casos para dar respuesta a preguntas relevantes en la investigación. Para la realización del proyecto se han escogido cuatro comunidades como casos de estudio, concretamente OpenOffice, Mozilla, NetBeans y Eclipse. La elección se basa en la familiarización con el software y el éxito de dichas comunidades, puesto que consideramos un factor importante el amplio número de usuarios que utilizan el software desarrollado por estas comunidades OSS.

Paso 3. Descripción de las comunidades. Para comprender mejor el contexto de las comunidades consideramos necesario conocer los siguientes aspectos: la procedencia de la comunidad y la descripción de la funcionalidad del software, el tipo de gobierno, la documentación disponible y los distintos tipos de infraestructuras que utilizan para trabajar y otros datos de interés como los tipos de licencias que emplean.

Paso 4. Definición de las preguntas de investigación. Formulamos una serie de preguntas teniendo en cuenta nuestro objetivo y basadas en estudios previos de características similares, para conocer el proceso de requisitos de cada comunidad. Estas preguntas se detallan a continuación:

1. *¿Cómo se obtienen o educen los requisitos del software?*
   Queremos saber si utilizan algún método de la Ingeniería de Requisitos tradicional para obtener los requisitos de software, puesto que no se encuentran muchos estudios sobre la obtención de requisitos en proyectos OSS.

2. *¿Cuáles son las principales fuentes de requisitos (desarrolladores, usuarios, otras fuentes)? ¿Se diferencian los participantes?*

   Conocer de dónde proceden los requisitos nos permite decidir qué factores participan en la educción de requisitos. En estudios previos [Scacchi, 2002][Heliö, 2004] se especifica que las principales fuentes de requisitos son los desarrolladores y usuarios, por lo que comprobaremos esta afirmación y si existen otras fuentes de requisitos.

3. *¿Los requisitos son claros o vagos, el desarrollador es consciente de las necesidades?*

   Con esta pregunta pretendemos observar un factor que influye en la calidad del software y que afecta al resto del proceso de desarrollo del software; si los requisitos no están bien definidos desde el principio, puede ocasionar problemas mayores en posteriores etapas del desarrollo.

4. *¿El proceso está formalizado o controlado?*

   Se ha planteado esta pregunta para comparar el proceso seguido en las comunidades OSS con el de la Ingeniería de Requisitos tradicional.

5. *¿De qué forma se toman las decisiones y resuelven los conflictos (acuerdo, decisión unilateral, diferentes versiones, etc.)? ¿Cómo se establecen las prioridades de los requisitos?*

   Dando respuesta a esta pregunta podemos obtener información sobre el gobierno y el tipo liderazgo que tiene la comunidad, además de su gestión.

6. *¿Cómo son los requisitos analizados y diseñados?*

   No hay estudios donde se analice esta actividad en OSS, por lo que consideramos importante plantear esta pregunta.

7. *¿Se documentan los requisitos y se clasifican, dónde?*

   En la Ingeniería de Requisitos tradicional la documentación es un factor importante en la comunicación entre los ingenieros de software y también con los clientes. En este estudio observaremos cómo se realiza la especificación de requisitos en OSS.

8. *¿Cómo se gestionan las peticiones de funcionalidades?*

   Con el fin de conocer el proceso de requisitos es importante observar de qué forma se comunican y educen los requisitos.

9. *¿Qué prácticas de la Ingeniería de Requisitos tradicional se observan? ¿Podemos identificar otras?*

Nuestro objetivo es comprender y determinar cómo se llevan a cabo los procesos y actividades relacionados con la Ingeniería de Requisitos en una comunidad OSS, por lo que se trata de una pregunta de relevancia que nos aportará más información sobre las diferencias y similitudes.

Paso 5. Análisis y discusión de resultados. A partir de las respuestas obtenidas de las preguntas realizadas analizaremos los resultados y comprobaremos las afirmaciones realizadas en el Capítulo 4 sobre el proceso de requisitos de las comunidades OSS.

Capítulo 4

DIFERENCIAS Y SIMILITUDES DEL PROCESO DE REQUISITOS DEL SOFTWARE TRADICIONAL Y OSS

El objetivo de este capítulo es determinar las diferencias y similitudes entre el proceso de requisitos de la comunidad OSS y el que establece la Ingeniería del Software tradicional, de modo que podamos comprobar determinadas afirmaciones en los proyectos OSS seleccionados. Para ello nos basaremos en estudios previos [Castro y Acuña, 2012][Scacchi et al., 2005][Scacchi, 2004][Scacchi, 2002].

Tal y como se describe cada una de las actividades del proceso de requisitos del software tradicional en el apartado 2.2 del Capítulo 2, Estado de la Cuestión, seguiremos ese mismo orden para realizar la comparación de forma metódica. En ninguno de los estudios analizados se incluyen actividades relacionados con la validación de los requisitos, lo que sugiere que las comunidades OSS no realizan la validación del mismo modo que la Ingeniería de Requisitos tradicional. Por ello, no se describe esta actividad.

4.1 Educción de Requisitos

Como se ha visto en el Capítulo 2, Estado de la Cuestión, por un lado, en la educción de requisitos del proceso de la Ingeniería del Software es el ingeniero quien debe educirlos e interpretarlos de otras personas, los usuarios o clientes, lo que complica la comprensión de los mismos y conlleva a conflictos que han de resolverse con el mutuo acuerdo de ambas partes. La educción de todos los requisitos sigue un proceso

formalizado donde la adición de nuevos requisitos pasa por un procedimiento controlado.

Por otro lado, en las comunidades OSS cualquier persona puede solicitar la adición de una funcionalidad, normalmente los mismos desarrolladores son los usuarios finales, por lo que el desarrollador tiene un buen entendimiento de lo que desea. Las ideas son compartidas e integradas y las decisiones sobre incluir funcionalidades y cambios son discutidas por la comunidad y los propietarios de los módulos, empleando para ello la realización de comentarios en los grupos de noticias.

La esencia es la misma, educir los requisitos y resolver los conflictos generados, y el resultado es el mismo, obtener una lista de requisitos. En ambos casos los requisitos nacen de necesidades específicas.

4.2 Análisis de Requisitos

En las comunidades OSS por lo general los desarrolladores se encargan de aceptar las peticiones de los usuarios, cuando no se resuelven los conflictos se crean diferentes versiones. En el análisis de los requisitos de la comunidad contribuyen un amplio número de desarrolladores, que son a la vez los usuarios del sistema software.

Mientras que en la Ingeniería de Requisitos tradicional los participantes en el análisis están bien diferenciados y es el ingeniero de software el que toma la decisión unilateral o consulta a los implicados para llegar a un acuerdo.

4.3 Especificación de Requisitos

En el caso del software propietario un contrato exige la especificación de los requisitos del software como entregable y es el ingeniero el encargado de especificar y formalizar los requisitos.

En sistemas de OSS no se suelen producir especificaciones formalizadas de requisitos, pues estos cambian continuamente y se discuten públicamente, por lo que se encuentran como mensajes en foros de discusión. No hay obligaciones contractuales, de modo que en la mayor parte de los casos no existe un documento de especificación de requisitos, solo una lista de tareas por hacer, donde solo se especifica el tipo de la misma.

En ambos casos existen requisitos que se describen, ya sea en documentos o en foros de discusión o correos electrónicos, y también se clasifican los requisitos.

4.4 Gestión de los Requisitos

Para facilitar el seguimiento y el control de los requisitos, en ambos casos se asigna una prioridad y un orden a los requisitos. En el caso del OSS se genera una lista de tareas a las que se asignan prioridades y niveles de dificultad, mientras que en el proceso de requisitos tradicional existe un procedimiento formalizado para la gestión de los requisitos.

En la Tabla 3 se resumen las diferencias y las similitudes del proceso de requisitos tradicional y OSS analizadas:

Diferencias		**Similitudes**
OSS	**Tradicional**	
Comienza con la idea de satisfacer unos requisitos.	Comienza con la definición y especificación de requisitos.	- La esencia de la educción de requisitos es la misma: identificar los requisitos. - La esencia del análisis de requisitos es la misma: resolver los conflictos generados. - Los requisitos nacen de necesidades específicas. - Los requisitos funcionales y no funcionales son documentados. - La esencia de la gestión de requisitos es la misma: se asignan prioridades a tareas o requisitos.
Los desarrolladores son a su vez usuarios finales.	Los participantes en este análisis están bien diferenciados ingeniero de software/usuarios finales.	
Requisitos claros.	Requisitos vagos.	
El desarrollador es consciente de sus necesidades.	El desarrollador no es consciente de las necesidades.	
Normalmente no existen documentos formales de especificación, los requisitos se encuentran distribuidos en foros.	Existen documentos formales de especificación de requisitos del software.	
Se afirman, analizan, especifican, negocian y priorizan con artefactos basados en la Web.	Se educen, analizan, especifican, negocian y priorizan en reuniones cara a cara.	

Tabla 3. Comparativa del Proceso de Requisitos Tradicional y OSS

Por tanto, podemos afirmar que el proceso de requisitos OSS tiene las siguientes características:

- Respecto a la Educción de Requisitos:
  - Los requisitos provienen de la idea de satisfacer las necesidades tanto de usuarios como desarrolladores.
  - Los desarrolladores son a su vez usuarios finales.
  - Los requisitos son claros y el desarrollador es consciente de sus necesidades.
  - No existe un proceso formalizado para la educción de requisitos, las ideas son compartidas y discutidas por la comunidad.
- Respecto al Análisis de Requisitos:
  - El número de desarrolladores no está definido y cualquier usuario puede participar y colaborar.
  - Las decisiones se toman por medio de discusiones entre los miembros de la comunidad.
- Respecto a la Especificación de Requisitos:
  - No existen documentos formales de especificación de requisitos, estos se encuentran distribuidos en infraestructuras online, como los foros.
- Respecto a la Gestión de Requisitos:
  - Se utilizan listas de tareas a las que se asignan prioridades.
  - La gestión se realiza por medio de infraestructuras online.

Capítulo 5

COMPARATIVA DEL PROCESO DE REQUISITOS EN EL DESARROLLO DE OPEN SOURCE SOFTWARE

En este capítulo se describen cada una de las comunidades OSS seleccionadas, sus respectivas infraestructuras y los datos obtenidos que consideramos relevantes en la investigación. Posteriormente con dichos datos se realiza un análisis para verificar las afirmaciones declaradas en el capítulo anterior.

5.1 Descripción de las Comunidades

En los siguientes apartados se detallarán los datos obtenidos en la investigación de cada una de las comunidades OSS, sus características y las respuestas a las preguntas planteadas sobre el proceso de requisitos y las infraestructuras que utilizan. En los Anexos se puede encontrar reflejado en las bitácoras el proceso seguido para la obtención de los datos (Anexo A) y más información detallada sobre el estudio realizado de cada comunidad (Anexo B).

5.1.1 OpenOffice

Apache OpenOffice, antes conocido como OpenOffice.org, es una alternativa gratuita para las herramientas ofimáticas [OpenOffice, 2014]. Consiste en una serie de software de oficina de código abierto que contiene procesador de textos, hoja de cálculo, presentación, gráficos, editor de fórmulas y aplicaciones de gestión de bases de datos. OpenOffice está disponible en muchos idiomas, funciona en todos los ordenadores

comunes dando soporte a distintos sistemas operativos, almacena los datos en ODF, el formato estándar abierto internacional, y es capaz de leer y escribir archivos en otros formatos, incluido el formato utilizado por los paquetes más habituales de ofimática. OpenOffice también es capaz de exportar archivos en formato PDF. OpenOffice ha apoyado las extensiones, de una manera similar a Mozilla Firefox, haciendo fácil añadir nuevas funciones a una instalación existente de OpenOffice.

Criterios de selección: Este proyecto fue seleccionado debido que ya se había trabajado con este software anteriormente y dado su éxito, actualmente celebra las más de 100.000.000 de descargas.

Búsqueda: No fue necesaria una búsqueda ya que conocía la aplicación. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: El conjunto de herramientas ofimáticas es descargable en distintos idiomas y de forma gratuita desde SourceForge o desde la página de descargas de la comunidad [OO Descarga].

Contacto: En su página web, existe una sección en la que se informa de todas las formas en las que se puede participar [OO Contacto].

En la Tabla 4 se muestran las características de la comunidad OpenOffice, concretamente los roles que podemos encontrar en la comunidad, el tipo de gobierno y la distribución de los miembros de la comunidad, la documentación existente, las principales infraestructuras, las licencias, los lenguajes de programación utilizados y otra información que consideramos relevante.

Características
Roles de la comunidad
Los contribuidores y desarrolladores se reparten las tareas en distintos proyectos [OO Gobierno]: - ***Poodling* PMC (PPMC):** es el grupo de miembros del PMC elegidos que supervisan el proyecto, eligen las aportaciones y votan para aprobar publicaciones. - **Dev:** incluye programadores de C++ o Java, especialistas de la interfaz de usuario, el rendimiento, etc. - **QA:** se encargan de las pruebas manuales y automatizadas. - **Traducción:** demuestran la capacidad de adaptación lingüística y cultural del producto. - **Documentación:** incluye la ayuda en el producto así como la documentación en la Web. - ***Support*:** se dedican al soporte en los foros. - **Marketing:** son los responsables de la publicidad, eventos, promociones, etc. - ***Committers*:** es el grupo de contribuidores elegidos por el PPMC, que puede adquirir cualquiera de los roles anteriores y pueden comprobar el código de los repositorios y de cualquier wiki directamente.
Gobierno de la comunidad
Comunidad con principios comunes y reglas formales: los miembros del comité PMC son los responsables de orientar el proyecto, pero este comité se compone de miembros escogidos por la comunidad y los usuarios; desarrolladores y contribuidores pueden formar parte en la toma de decisiones.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 4.530 usuarios y 94 contribuidores que se encuentran distribuidos en gran medida. La mayoría de usuarios y contribuidores proceden de Europa y Estados Unidos entre otros países como India o Brasil.
Documentación
Para usuarios: proporciona guías de instalación y guías de usuario de cada una de las herramientas y una sección de FAQs, HOWTOs y tutoriales. Para desarrolladores: ofrece guías sobre BASIC el lenguaje de programación y el código de Apache OpenOffice [OO Documentación].

Tabla 4. Características de la comunidad OpenOffice

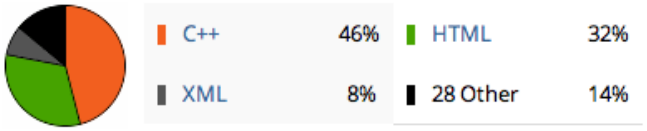
Características
Infraestructuras
- **The Apache OpenOffice Wiki:** donde se publican noticias, información sobre actualizaciones, cómo participar en la comunidad por medio de las listas de correos o foros y documentación [OO Wiki]. - **IssueTraker y Bugzilla:** un lugar para informar o buscar errores, es necesario registrarse para poder publicar errores [OO Issues] [OO Bugzilla]. - **The Apache OpenOffice Community Forum:** recomendado para todos los usuarios de OpenOffice, proporciona ayuda a los usuarios por parte de voluntarios de la comunidad [OO Foro]. - **Listas de Correos:** son listas públicas clasificadas en distintas necesidades, es necesario registrarse para poder publicar mensajes. Para gestionar estos mensajes utilizan distintas plataformas como Markmail, Gmane, Mail Archive o los propios archivos de Apache, que permiten realizar búsquedas de mensajes por hilos de mensajes, fechas o temas [OO Listas de correos]. Para facilitar la búsqueda se pueden utilizar ciertas etiquetas que especifican el asunto del mensaje. - **Redes Sociales:** además de las infraestructuras mencionadas OpenOffice está presente en Google+, Indenti.ca, Twitter, Hashtags, Facebook y XING, donde publican las últimas noticias.
Licencias
Apache License 2.0 Public Document License (PDL) GNU Lesser General Public License (LGPL) v3 Creative Commons Attribution License (“Attribution-NoDervs 2.5”)
Lenguajes utilizados

Otros
No se han encontrado más características relevantes.

Tabla 4. Características de la comunidad OpenOffice (Continuación)

Después de obtener las características de la comunidad, se lleva a cabo la investigación a fondo de los foros y las listas de correo, con el fin de encontrar y analizar las actividades del proceso de requisitos. A continuación, se detallan las respuestas observadas para la serie de preguntas que se han planteado.

1. *¿Cómo se obtienen o educen los requisitos del software?*

   En la comunidad OpenOffice no hemos encontrado ningún método de obtención de requisitos tal y como se estudia en la Ingeniería de Requisitos tradicional. Los requisitos proceden de las necesidades directas de los desarrolladores o de peticiones o sugerencias de los usuarios finales.

Solo se encontró un correo donde se preguntaba abiertamente ideas para la próxima versión del software [OO Brainstorming].

2. *¿Cuáles son las principales fuentes de requisitos (desarrolladores, usuarios, otras fuentes)? ¿Se diferencian los participantes?*

   Según las participaciones observadas en el estudio de los foros y de las listas de correos, la mayor fuente de requisitos son las que plantean los desarrolladores y las sugerencias de los usuarios finales.

   No es fácil distinguir entre usuarios y desarrolladores en los foros o en las listas de correos, puesto que ambos participan en las discusiones y no hay ningún identificador en las infraestructuras que nos permita distinguirlos. En la lista de desarrolladores la mayor parte de participantes son contribuidores o desarrolladores, mientras que en las listas de usuarios es posible distinguir que las preguntas son planteadas por usuarios y las respuestas vienen dadas por desarrolladores.

3. *¿Los requisitos son claros o vagos, el desarrollador es consciente de las necesidades?*

   En muchas ocasiones es el propio desarrollador el que plantea el requisito y lo implementa por lo que es plenamente consciente de sus necesidades, pero en ocasiones podemos encontrar discusiones para conocer más detalles de las peticiones.

4. *¿El proceso está formalizado o controlado?*

   En esta comunidad se explica el proceso que pueden seguir los desarrolladores, pero no es obligatorio y tampoco está controlado de ningún modo. Sí que existe un equipo que supervisa la calidad del software y lo que se añadirá en la próxima versión, pero no se tiene ningún compromiso.

5. *¿De qué forma se toman las decisiones y resuelven los conflictos (acuerdo, decisión unilateral, diferentes versiones, etc.)? ¿Cómo se establecen las prioridades de los requisitos?*

   El PMC (*Project Management Committee*) es el comité responsable del proyecto y decide qué hacer o qué dirección tomar. Aunque hay subsistemas que tienen voluntarios como administradores. Se puede formar parte del comité por meritocracia: si se envían parches de calidad para mejorar el software, si se añade documentación a la Wiki o a la página web, si se demuestra que es capaz

de debatir en las listas de correo o si se es reconocido entre los miembros de la comunidad [OO PMC FAQs].

En proyectos de Apache habitualmente no se utiliza el sistema de votos, se trabaja con las siguientes técnicas de consenso para tomar decisiones [OO Community]:

Consenso vago asume que la comunidad está de acuerdo y se tiene vía libre para trabajar, no es necesario discutir o aprobar la propuesta, lo que evita el tráfico de mensajes innecesarios. Al utilizar herramientas como Subversion siempre es posible volver a una versión anterior si surge alguna discrepancia. Cuando se realiza la propuesta se puede declarar un periodo de tiempo, por ejemplo 72 horas, antes de comenzar su implementación, y si en ese periodo nadie se opone se puede proceder con la propuesta.

Creación del consenso se utiliza en el caso de que sea necesario debatir opciones, donde también se desee evitar mensajes innecesarios que tengan que leer toda la comunidad. Para expresar el apoyo o no se usa la siguiente notación:

- +1 significa "Estoy de acuerdo y colaboraré".
- +0 significa "Estoy de acuerdo pero probablemente no ayude, así que mi opinión no es tan importante".
- -0 significa "No estoy de acuerdo, pero no ofrezco otra alternativa así que mi opinión no es tan importante".
- -1 significa "No estoy de acuerdo y ofrezco otra alternativa en la que puedo colaborar".

En ocasiones el valor de los números varía para expresar el apoyo de modo más informal, como por ejemplo +0,5 o +1000. La razón de esta notación es que basta con sumar para obtener un consenso o no. Si se obtiene un consenso, se puede proceder con el consenso vago.

Voto se realiza para tomar decisiones formales o por razones legales, como por ejemplo votar a un nuevo miembro del comité. Cada usuario tiene un solo voto. La notación utilizada en la creación del consenso es la misma para la votación, la diferencia está en que el asunto contiene "[Vote]".

Si se trata de una contribución puntual no se le atribuye una prioridad, pero en el caso de los proyectos o la corrección de errores asignados a un grupo de desarrolladores, son estos los que deciden la prioridad de cada tarea.

6. *¿Cómo son los requisitos analizados y diseñados?*

   En esta comunidad no se ha encontrado información sobre el análisis o el diseño de los requisitos en ninguna de las infraestructuras estudiadas. Tampoco se explica cómo se realiza el análisis o si se utiliza alguna aplicación para el diseño. Por lo que desconocemos si los desarrolladores realizan un análisis y un diseño antes de comenzar con la implementación y no lo hacen de forma pública.

7. *¿Se documentan los requisitos y se clasifican, dónde?*

   No existe una documentación formal de los requisitos como ocurre en la Ingeniería de Requisitos tradicional, pero podemos encontrar la especificación de los requisitos definida en infraestructuras online, como son los foros o las listas de correos.

8. *¿Cómo se gestionan las peticiones de funcionalidades?*

   Habitualmente en el caso de la corrección de errores se utiliza la herramienta Bugzilla, lo que permite gestionar de forma más cómoda las peticiones, sino se comunican por medio de canales como el foro o la lista de correos. De forma voluntaria un desarrollador se hace cargo de la petición e informa de que realizará la tarea y si necesita ayuda o no, para que no asignen a otra persona la misma tarea. En el caso de la corrección de errores, se asigna un identificador numérico a la petición y se comprueba si existen duplicados o tareas relacionadas. Una vez implementadas se hacen públicas en un repositorio como Subversion, de modo que el equipo de calidad compruebe el código y dé su visto bueno. Si no cumple con la calidad esperada, se vuelve a informar al responsable. Todo este proceso al tratarse de OSS es voluntario, de modo que en ocasiones una tarea se abandona y queda pendiente hasta que otra persona la retoma o se descarta.

9. *¿Qué prácticas de la Ingeniería de Requisitos tradicional se observan? ¿Podemos identificar otras?*

   En principio no se han observado prácticas de la Ingeniería de Requisitos tradicional en esta comunidad. La mayor parte de las actividades del proceso de requisitos se realiza mediante infraestructuras online, a través de Internet y no cara a cara. Solo hemos podido encontrar una técnica para la educción de requisitos, concretamente el *brainstorming* por parte de usuarios y desarrolladores.

5.1.2 Mozilla

Mozilla es una organización no lucrativa dedicada a mantener viva y accesible la Web [Mozilla, 2014]. Se trata de una comunidad global de usuarios, colaboradores y desarrolladores que trabajan en la innovación. El proyecto Mozilla utiliza un enfoque basado en la comunidad para crear un software de código abierto de ámbito mundial y el desarrollo de nuevos tipos de actividades centradas en la colaboración. Entre sus proyectos destaca el navegador Mozilla Firefox, el tercer navegador más usado actualmente y en el que nos centraremos para este estudio. Otros proyectos también muy populares son Thunderbird, Bugzilla o Firebug.

Criterios de selección: Este proyecto fue escogido porque los proyectos de Mozilla son unos de los más activos y populares y tiene una de las comunidades más grandes.

Búsqueda: No fue necesaria una búsqueda ya que conocía la aplicación. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: Los productos de Mozilla se pueden descargar de forma gratuita desde su página oficial [Mz Descarga].

Contacto: Existen lugares físicos distribuidos en el mundo donde se pueden encontrar las oficinas de Mozilla, donde la gente se reúne para colaborar y construir la Web. Pero también nos podemos poner en contacto con la comunidad más cercana a nosotros aprovechando las infraestructuras online disponibles [Mz Contacto].

En la Tabla 5 se muestran las características de la comunidad Mozilla, concretamente los roles que podemos encontrar en la comunidad, el tipo de gobierno y la distribución de los miembros de la comunidad, la documentación existente, las principales infraestructuras, las licencias, los lenguajes de programación utilizados y otra información que consideramos relevante.

Características
Roles de la comunidad
El proyecto Mozilla está gobernado por un equipo de dirección virtual, compuesto de personas con distintos roles de liderazgo escogidos por meritocracia, es decir por sus actividades y aportaciones de calidad a la comunidad. Los distintos roles de liderazgo que podemos encontrar son [Mz Gobierno]: - **Propietarios de Módulo o Peers:** son responsables de dirigir el desarrollo de un módulo de código o una actividad comunitaria. - **Antiguos Propietarios de Módulo:** se aplica a los propietarios de módulo que se trasladan a otras funciones dentro de Mozilla o que ya no están activos. - **Super-Revisores:** son un grupo de hackers que revisan el código y comprueban que cumple las guías de codificación de Mozilla. - ***Release Drivers*:** se encargan de la gestión de los proyectos en el lanzamiento de versiones. - **Propietarios de componentes de Bugzilla:** puede ser la misma persona que el propietario del módulo, se encarga de revisar los informes de errores regularmente, reasigna los errores para corregir y se asegura de hacer un seguimiento de su progreso y de que existen casos de prueba. - ***Mozilla Reps*:** son representantes de Mozilla en su país o región y que se dedican a educar a la gente para apoyar y contribuir a Mozilla. - ***Mozilla Reps Mentors*:** proveen orientación y son responsables de revisar los informes mensuales, además de las solicitudes de presupuesto. - ***Mozilla Reps Council*:** el consejo está formado por 9 miembros del programa y 7 voluntarios, que supervisan la gestión y las finanzas del programa. - **Administradores:** son responsables de mantener las vías de contribución, además realizan tareas de reconocimiento, educación y métricas para mantener dichas vías. - ***Ultimate Decision-Makers*:** en casos de conflicto tienen la última palabra, Brendan Eich en las disputas técnicas y Mitchell Baker en cualquier disputa no técnica.
Gobierno de la comunidad
Organización o dictador benevolente: la comunidad tiene una organización jerárquica que se repite en cada subproyecto, donde los usuarios, desarrolladores y colaboradores participan en las discusiones. Pero las decisiones finales son tomadas por el líder del proyecto o el propietario del módulo.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 12.945 usuarios y 2.959 contribuidores que se encuentran distribuidos en gran medida. La mayoría de usuarios y contribuidores proceden de Europa y Estados Unidos entre otros países como India o Japón.
Documentación
En la página web de Mozilla no es fácil acceder a la documentación y la mayor parte son guías para el usuario o tutoriales [Mz Documentación].

Tabla 5. Características de la comunidad Mozilla

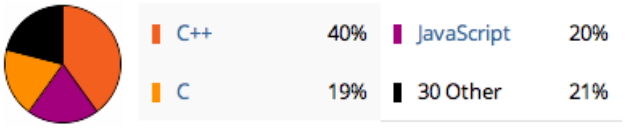
Características
Infraestructuras
- **MozilaWiki:** donde se publican las actividades de la comunidad y el desarrollo de los proyectos, noticias e información sobre actualizaciones [Mz Wiki]. - **Bugzilla:** un lugar para informar o buscar errores, es necesario registrarse para poder publicar errores [Mz Bugzilla]. - **Mozilla Forums:** está organizado como las listas de correos o grupos de noticias, clasificadas en distintas categorías. Los mensajes están archivados en Google Groups y están gestionados por Giganews NewsGroups [Mz Foro]. - **Mozilla's Chat Server:** dispone de un sistema de chat para comunicarse en tiempo real con otros miembros de la comunidad, es necesario estar registrado [Mz Chat]. - **Mozilla Developer Network (MDN):** es una red de desarrolladores que facilita toda la información para hacer posible la colaboración en los distintos proyectos de Mozilla [Mz MDN].
Licencias
Mozilla Public License MPL v2.0
Lenguajes utilizados

Otros
Recibe financiación de algunas empresas, entre ellas Google.

Tabla 5. Características de la comunidad Mozilla (Continuación)

Después de obtener las características de la comunidad, se lleva a cabo la investigación a fondo de los foros y las listas de correo, con el fin de encontrar y analizar las actividades del proceso de requisitos. A continuación, se detallan las respuestas observadas para la serie de preguntas que se han planteado.

1. *¿Cómo se obtienen o educen los requisitos del software?*

   En este proyecto no hemos encontrado ningún método de obtención de requisitos tal y como se estudia en la Ingeniería de Requisitos tradicional. Los requisitos proceden principalmente de las necesidades directas de los desarrolladores y de peticiones o sugerencias de los usuarios finales.

2. *¿Cuáles son las principales fuentes de requisitos (desarrolladores, clientes, usuarios)? ¿Se diferencian los participantes?*

   Según las participaciones observadas en el estudio de los foros y de las listas de correos la mayor fuente de requisitos son las que plantean los desarrolladores, aunque también podemos encontrar sugerencias de los usuarios finales.

   No es fácil distinguir entre usuarios y desarrolladores en los foros o en las listas de correos, puesto que ambos participan en las discusiones y no hay ningún

identificador en las infraestructuras que nos permita distinguirlos. Especialmente en la lista de correos de los usuarios parece ser que al tratarse de una gran comunidad los usuarios se resuelven dudas entre sí.

3. *¿Los requisitos son claros o vagos, el desarrollador es consciente de las necesidades?*

   En muchas ocasiones es el propio desarrollador el que plantea el requisito y lo implementa por lo que es plenamente consciente de sus necesidades, pero en ocasiones encontramos discusiones para conocer más detalles de las peticiones. Podemos encontrar en la comunidad de Mozilla discusiones sobre la necesidad de una funcionalidad antes de ser aceptada en el repositorio de Bugzilla.

4. *¿El proceso está formalizado o controlado?*

   En esta comunidad se explica el proceso que pueden seguir los desarrolladores, pero no es obligatorio y tampoco está controlado de ningún modo. Sí que existe un equipo que supervisa la calidad del software y lo que se añadirá en la próxima versión, pero no se tiene ningún compromiso.

5. *¿De qué forma se toman las decisiones y resuelven los conflictos (acuerdo, decisión unilateral, diferentes versiones, etc.)? ¿Cómo se establecen las prioridades de los requisitos?*

   En Mozilla existen distintos módulos de desarrollo que tienen proyectos, cada uno de ellos con su propio administrador o propietario que se dedica a orientar el proyecto. Existe un sistema de votos similar al explicado anteriormente en la comunidad de OpenOffice.

6. *¿Cómo son los requisitos analizados y diseñados?*

   En esta comunidad no se ha encontrado información sobre el análisis o el diseño de los requisitos en ninguna de las infraestructuras estudiadas. Tampoco se explica cómo se realiza el análisis o si se utiliza alguna aplicación para el diseño. Por lo que desconocemos si los desarrolladores realizan un análisis y un diseño antes de comenzar con la implementación y no lo hacen de forma pública.

7. *¿Se documentan los requisitos y se clasifican, dónde?*

   No existe una documentación formal de los requisitos como ocurre en la Ingeniería de Requisitos tradicional, pero podemos encontrar la especificación de los requisitos definida en infraestructuras online, como son los foros o las listas de correos y podemos encontrar su desarrollo en la red MDN.

8. *¿Cómo se gestionan las peticiones de funcionalidades?*

Después de las discusiones en las listas de correo o foros, si se acepta la propuesta pasa a ser gestionada en el entrono de Bugzilla [Mz Proceso].

9. *¿Qué prácticas de la Ingeniería de Requisitos tradicional se observan? ¿Podemos identificar otras?*

   En principio no se han observado prácticas de la Ingeniería de Requisitos tradicional en esta comunidad. La mayor parte de las actividades del proceso de requisitos se realiza mediante infraestructuras online, a través de Internet y no cara a cara. La relación que podemos encontrar está en el proceso de gestión con el control de versiones, que a diferencia de las empresas en OSS se realiza con más frecuencia y de forma menos controlada, pero esto no forma parte de nuestro estudio puesto que ya es parte del proceso de implementación.

5.1.3 NetBeans

NetBeans IDE es un entorno de desarrollo disponible para Windows, Mac, Linux y Solaris [NetBeans, 2014]. El proyecto NetBeans consiste en un IDE (*Integrated Development Environment*) de código abierto y una plataforma de aplicaciones que permiten a los desarrolladores crear rápidamente páginas web, software para empresas o particulares y aplicaciones móviles utilizando la plataforma de Java, así como PHP, JavaScript y Ajax, Groovy y Grails, C y C++. El proyecto de NetBeans está apoyado por una comunidad de desarrolladores dinámica y ofrece una amplia documentación y recursos, así como una variada selección de plugins de terceros.

Criterios de selección: Este proyecto fue escogido porque junto con Eclipse es una de las herramientas OSS utilizadas para el desarrollo de software y ya estaba familiarizada con la plataforma.

Búsqueda: No fue necesaria una búsqueda ya que conocía la aplicación. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: El entorno de desarrollo software y otros plugins se pueden descargar de forma gratuita desde SourceForge o desde la página de descargas de la comunidad [Nb Descarga].

Contacto: En su página web, existe una sección en la que se informa de todas las formas en las que se puede participar [Nb Contacto].

En la Tabla 6 se muestran las características de la comunidad NetBeans, concretamente los roles que podemos encontrar en la comunidad, el tipo de gobierno y la distribución de los miembros de la comunidad, la documentación existente, las principales infraestructuras, las licencias, los lenguajes de programación utilizados y otra información que consideramos relevante.

Características
Roles de la comunidad
Los distintos roles que podemos encontrar en NetBeans son [Nb Gobierno]: - **Usuarios:** son las personas que usan la plataforma o el IDE, informan de errores y hacen peticiones o sugerencias. - **Colaboradores:** son los miembros que contribuyen a netbeans.org pero que no tienen acceso de escritura al código fuente principal, pueden realizar aportaciones de parches, código nuevo o informes de errores, también pueden añadir contenidos como artículos, FAQs o pantallazos. Según sus aportaciones pueden llegar a ser desarrolladores o encargados. - **Desarrolladores:** tienen acceso de escritura al código fuente principal. - **Encargados:** es el encargado de juntar los parches de los contribuyentes, las correcciones de errores y el nuevo código, asegurándose de que estas aportaciones son estables y cumplen con la calidad adecuada. Cada módulo tiene un encargado que tiene permisos de registro y gestiona a un grupo de desarrolladores. - **Junta Directiva:** está integrada por tres miembros, uno asignado por Oracle y otros dos escogidos por la comunidad. La Junta existe como último recurso para resolver disputas. Actualmente la componen: Ashwin Rao (Oracle), Tim Boudreau y Zoran Sevarac [Nb Contribuidores].
Gobierno de la comunidad
Líder + organización formal: NetBeans tiene una Junta Directiva que se involucra en la toma de decisiones, pero los usuarios, desarrolladores y colaboradores forman una parte activa en la toma de decisiones.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 939 usuarios y 668 contribuidores que se encuentran distribuidos en gran medida. La mayoría de usuarios y contribuidores proceden de Europa y Estados Unidos entre otros países como Brasil o Japón.
Documentación
Junto con las guías de instalación y video tutoriales, contiene guías de usuario sobre cómo desarrollar aplicaciones en NetBeans en los distintos lenguajes de programación disponibles en la plataforma, además de un repositorio de versiones anteriores. También se ofrece una sección de FAQs para usuarios y desarrolladores en la wiki [Nb Documentación].

Tabla 6. Características de la comunidad NetBeans

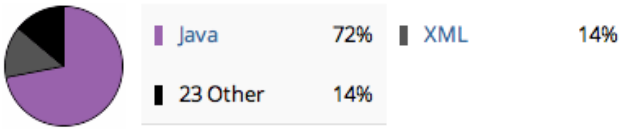
Características
Infraestructuras
- **NetBeans Wiki:** donde se publica información sobre actualizaciones, cómo participar en la comunidad, FAQs y documentación [Nb Wiki]. - **Issues y Bugzilla:** un lugar para informar o buscar errores, es necesario registrarse para poder publicar errores [Nb Issues]. - **NetBeans Forums:** está separado en distintas categorías según el lenguaje en el que se desarrolle [Nb Foro]. - **Listas de Correos:** son listas públicas clasificadas en distintas necesidades, es necesario registrarse para poder publicar mensajes [Nb Listas de correos]. - **NetBeans User Chat:** dispone de un sistema de chat para comunicarse en tiempo real con otros miembros de la comunidad [Nb Chat].
Licencias
Common Development and Distribution License (CDDL) v1.0 GNU General Public License (GPL) v2.0
Lenguajes utilizados

Otros
Oracle apoya NetBeans como un proyecto de código abierto y paga los salarios de los desarrolladores que trabajan en ello. Project Kenai mantiene la infraestructura del sitio netbeans.org.

Tabla 6. Características de la comunidad NetBeans (Continuación)

Después de obtener las características de la comunidad, se lleva a cabo la investigación a fondo de los foros y las listas de correo, con el fin de encontrar y analizar las actividades del proceso de requisitos. A continuación, se detallan las respuestas observadas para la serie de preguntas que se han planteado.

1. *¿Cómo se obtienen o educen los requisitos del software?*

   En esta comunidad no hemos encontrado ningún método de obtención de requisitos tal y como se estudia en la Ingeniería de Requisitos tradicional. Los requisitos proceden de las necesidades directas de los desarrolladores y de peticiones o sugerencias de los usuarios finales.

2. *¿Cuáles son las principales fuentes de requisitos? ¿Se diferencian los participantes?*

   Según las participaciones observadas en el estudio de los foros y de las listas de correos casi no hay actividad, la mayor parte de las intervenciones son dudas de usuarios, pero las propuestas suelen provenir de desarrolladores.

No es fácil distinguir entre usuarios y desarrolladores en los foros o en las listas de correos, puesto que ambos participan en las discusiones y no hay ningún identificador en las infraestructuras que nos permita distinguirlos.

3. *¿Los requisitos son claros o vagos, el desarrollador es consciente de las necesidades?*

   No hemos encontrado discusiones por requisitos, pero encontramos discusiones sobre la implementación de código, en las cuales se hacen muchas referencias a la documentación en la wiki o a las librerías existentes.

4. *¿El proceso está formalizado o controlado?*

   En esta comunidad no se detalla el proceso que deben seguir los desarrolladores y tampoco está controlado de ningún modo solo es gestionado con Bugzilla.

5. *¿De qué forma se toman las decisiones y resuelven los conflictos (acuerdo, decisión unilateral, diferentes versiones, etc.)? ¿Cómo se establecen las prioridades de los requisitos?*

   La estructura del gobierno de NetBeans es muy plana, las decisiones se toman normalmente por consenso en debates públicos en listas de correos, antes que en votaciones. Habitualmente el consenso se llega entre las personas interesadas o afectadas, en el caso de haber un conflicto existe una junta directiva. Esta junta directiva está compuesta por tres personas, dos son nombradas por la comunidad y una tercera es asignada por Oracle.

6. *¿Cómo son los requisitos analizados y diseñados?*

   En esta comunidad no se ha encontrado información sobre el análisis o el diseño de los requisitos en ninguna de las infraestructuras estudiadas. Tampoco se explica cómo se realiza el análisis o si se utiliza alguna aplicación para el diseño. Por lo que desconocemos si los desarrolladores realizan un análisis y un diseño antes de comenzar con la implementación y no lo hacen de forma pública.

7. *¿Se documentan los requisitos y se clasifican, dónde?*

   No existe una documentación formal de los requisitos como ocurre en la Ingeniería de Requisitos tradicional, pero podemos encontrar la especificación de los requisitos definida en infraestructuras online, como son los foros o las listas de correos y existe una gran documentación sobre plugins y librerías implementadas en la wiki o en el javadoc.

8. *¿Cómo se gestionan las peticiones de funcionalidades?*

Habitualmente sigue el mismo proceso que la corrección de errores que se utiliza en la herramienta Bugzilla [Nb Proceso].

9. *¿Qué prácticas de la Ingeniería de Requisitos tradicional se observan? ¿Podemos identificar otras?*

   En principio no se han observado prácticas de la Ingeniería de Requisitos tradicional en esta comunidad. La mayor parte de las actividades del proceso de requisitos se realiza mediante infraestructuras online, a través de Internet y no cara a cara.

5.1.4 Eclipse

Eclipse es una comunidad de individuos y organizaciones que colaboran en el comercio de OSS [Eclipse, 2014]. Sus proyectos se centran en la construcción de una plataforma de desarrollo abierta formada por marcos extensibles y herramientas en tiempos de ejecución para la construcción, despliegue y gestión de software a través del ciclo de vida. La Fundación Eclipse es una corporación sin fines de lucro que alberga los proyectos de Eclipse y ayuda a desarrollar tanto la comunidad de código abierto como un ecosistema de productos y servicios complementarios. La plataforma Eclipse es una base genérica para un IDE, es decir, un entorno de desarrollo integrado con un sistema de plug-in extensible. Eclipse puede ser utilizado, por defecto, para desarrollar Java pero tiene plugins para apoyar otros lenguajes como: Ada, C, C++, COBOL, Perl, PHP, Python, R, Ruby (incluyendo el framework Ruby on Rails), Scala, Clojure, Groovy y Esquema. Puede crear proyectos genéricos, editar archivos en un editor de texto genérico y compartir los proyectos o archivos con un servidor de CVS (Sistema de Versiones Concurrentes).

Criterios de selección: Este proyecto fue escogido porque junto con Netbeans es otra herramienta OSS utilizada para el desarrollo de software con gran éxito.

Búsqueda: No fue necesaria una búsqueda ya que conocía la aplicación. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: El entorno de desarrollo software y otros plugins se pueden descargar de forma gratuita desde SourceForge o desde la página de descargas de la comunidad [Ec Descarga].

Contacto: En su página web existe una sección en la que se informa de todas las formas en las que se puede participar [Ec Contacto].

En la Tabla 7 se muestran las características de la comunidad Eclipse, concretamente los roles que podemos encontrar en la comunidad, el tipo de gobierno y la distribución de los miembros de la comunidad, la documentación existente, las principales infraestructuras, las licencias, los lenguajes de programación utilizados y otra información que consideramos relevante.

Características
Roles de la comunidad
Los proyectos en Eclipse tienen una jerarquía bien definida en la que podemos encontrar los siguientes roles [Ec Gobierno]: - **EMO (*Eclipse Managment Organization*):** está formado por un Staff de la fundación, el Consejo y Director Ejecutivo (ED) el cual tiene la autoridad en la toma de decisiones [Ec Directores]. - **Colaboradores:** es el grupo de desarrolladores contribuidores que realizan aportaciones al desarrollo de distintos proyectos de Eclipse. - **Usuarios:** todos aquellos que utilizan las herramientas disponibles de Eclipse. - **Adaptadores:** son los desarrolladores de plugins.
Gobierno de la comunidad
Organización o dictador benevolente: tiene una dirección ejecutiva que toma las decisiones sobre la orientación de los proyectos, aunque los usuarios, desarrolladores y colaboradores toman una parte activa en las discusiones.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 2.841 usuarios y 441 contribuidores que se encuentran distribuidos en gran medida. La mayoría de usuarios y contribuidores proceden de Europa y Estados Unidos entre otros países como China o Japón.
Documentación
Podemos encontrar guías de usuario de cada una de las versiones de Eclipse [Ec Documentación] y FAQs en la wiki.

Tabla 7. Características de la comunidad Eclipse

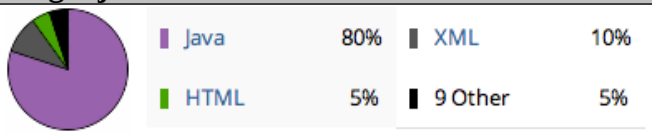
Características
Infraestructuras
- **Eclipsepedia:** donde se publican FAQs, información sobre actualizaciones, cómo participar en la comunidad y documentación [Ec Wiki]. - **Bugzilla:** un lugar para informar o buscar errores, es necesario registrarse para poder publicar errores [Ec Bugzilla]. - **Eclipse Community Forums:** está organizado según los distintos lenguajes de desarrollo del entorno de Eclipse y los distintos proyectos en desarrollo [Ec Foro]. - **Eclipse Mailing Lists:** son listas públicas clasificadas en proyectos, no están ordenadas por lo que no se aconseja para los usuarios [Ec Listas de correos].
Licencias
CPL-1.0 y EPL-1.0
Lenguajes utilizados

Otros
No se han encontrado más características relevantes.

Tabla 7. Características de la comunidad Eclipse (Continuación)

Después de obtener las características de la comunidad, se lleva a cabo la investigación a fondo de los foros y las listas de correo, con el fin de encontrar y analizar las actividades del proceso de requisitos. A continuación, se detallan las respuestas observadas para la serie de preguntas que se han planteado.

1. *¿Cómo se obtienen o educen los requisitos del software?*

   En la comunidad de Eclipse no hemos encontrado ningún método de obtención de requisitos tal y como se estudia en la Ingeniería de Requisitos tradicional. Los requisitos proceden de las necesidades directas de los desarrolladores o de peticiones o sugerencias de los usuarios finales.

2. *¿Cuáles son las principales fuentes de requisitos? ¿Se diferencian los participantes?*

   Según las participaciones observadas en el estudio de los foros y de las listas de correos la mayor fuente de requisitos son las que plantean los desarrolladores y las sugerencias de los usuarios finales.

No es fácil distinguir entre usuarios y desarrolladores en los foros o en las listas de correos, puesto que ambos participan en las discusiones y no hay ningún identificador en las infraestructuras que nos permita distinguirlos. Aunque en la descripción de los proyectos encontramos la lista de contribuidores que coincide con la persona que originalmente propone el proyecto.

3. *¿Los requisitos son claros o vagos, el desarrollador es consciente de las necesidades?*

   En muchas ocasiones es el propio desarrollador el que plantea el requisito y lo implementa por lo que es plenamente consciente de sus necesidades, pero en ocasiones podemos encontrar discusiones para conocer más detalles de las peticiones.

4. *¿El proceso está formalizado o controlado?*

   En esta comunidad se explica el proceso que deben seguir los desarrolladores; la junta directiva o los mentores de cada proyecto se encargan de supervisar cada propuesta y el avance del proyecto.

5. *¿De qué forma se toman las decisiones y resuelven los conflictos (acuerdo, decisión unilateral, diferentes versiones, etc.)? ¿Cómo se establecen las prioridades de los requisitos?*

   En Eclipse los proyectos tienen una jerarquía con responsables y mentores, pero es el EMO (*Eclipse Managment Organization*) el que se encarga de la toma de decisiones y de la supervisión de los proyectos.

6. *¿Cómo son los requisitos analizados y diseñados?*

   Para esta comunidad en la definición de los proyectos se puede encontrar detallado el alcance de los mismos, pero no se explica cómo se realiza el análisis o si se utiliza alguna aplicación para el diseño. Por lo que desconocemos si los desarrolladores realizan un análisis y un diseño antes de comenzar con la implementación y no lo hacen de forma pública.

7. *¿Se documentan los requisitos y se clasifican, dónde?*

   Cada proyecto tiene su página web, su lista de correos y foro, donde se puede encontrar la especificación de requisitos de modo informal.

8. *¿Cómo se gestionan las peticiones de funcionalidades?*

   Hemos observado que las peticiones se tratan en las infraestructuras disponibles y normalmente se gestionan con Bugzilla. Pero cada proyecto tiene su propio sistema [Ec Proceso].

9. *¿Qué prácticas de la Ingeniería de Requisitos tradicional se observan? ¿Podemos identificar otras?*

   En principio no se han observado prácticas de la Ingeniería de Requisitos tradicional en esta comunidad. La mayor parte de las actividades del proceso de requisitos se realiza mediante infraestructuras online, a través de Internet y no cara a cara. Lo único que pudimos encontrar fue un prototipo junto con una encuesta para la nueva interfaz de la página web, pero desconocemos si esto se aplica a otros proyectos.

5.2 Análisis de los Resultados

En este apartado se comparan los resultados obtenidos en el apartado anterior y se realiza un análisis global de los mismos verificando las afirmaciones planteadas en el Capítulo 4. A continuación, se muestran los resultados de las cuatro comunidades estudiadas en tablas comparativas, el tipo de liderazgo y las prácticas de trabajo observadas, además de las características del proceso de requisitos de cada una.

5.2.1 Resultados de los Tipos de Liderazgo

El tipo de liderazgo es un factor importante que influye en la gestión de los requisitos de una comunidad, ya que puede afectar directamente a los resultados de la obtención, análisis y especificación de los requisitos.

En la Tabla 8 podemos observar que aún dado el reducido número de casos de estudio hay una variedad de tipos de liderazgo. En OpenOffice encontramos una "Comunidad con principios comunes y reglas formales", mientras que en Mozilla y Eclipse tienen un liderazgo del tipo "Organización o dictador benevolente", tal y como se explicó en el apartado 2.4 según la clasificación dada por [Capra et al., 2008]. Y en la comunidad de NetBeans es del tipo "Líder + Organización formal".

Comunidad	Liderazgo	Observaciones
OpenOffice	Comunidad con principios comunes y reglas formales	El comité PMC es el responsable de orientar el proyecto, pero este comité se compone de miembros escogidos por la comunidad y los usuarios; desarrolladores y contribuidores pueden formar parte en la toma de decisiones.
Mozilla	Organización o dictador benevolente	La comunidad tiene una organización jerárquica que se repite en cada subproyecto, donde los usuarios, desarrolladores y colaboradores participan en las discusiones. Pero las decisiones finales son tomadas por el líder del proyecto o el propietario del módulo.
NetBeans	Líder + Organización formal	NetBeans tiene una junta directiva que se involucra en la toma de decisiones, pero los usuarios, desarrolladores y colaboradores forman una parte activa en la toma de decisiones.
Eclipse	Organización o dictador benevolente	Tiene una dirección ejecutiva que toma las decisiones sobre la orientación de los proyectos, aunque los usuarios, desarrolladores y colaboradores se incluyen en las discusiones.

Tabla 8. Resultados de los tipos de liderazgo

En las comunidades de Mozilla y Eclipse podemos observar claramente como la fundación que las mantiene trabaja para mejorar tanto el software que desarrollan como la comunidad que lo soporta. En Mozilla se dispone de una gran cantidad de infraestructuras y herramientas para los desarrolladores dirigidas por MDN (*Mozilla Developers Network*) y en Eclipse encontramos una detallada documentación sobre procesos. En ambas comunidades se observa una jerarquía que establece el estándar de calidad de sus productos.

En el caso de la comunidad OpenOffice o NetBeans existe un comité o una junta directiva que orienta los proyectos y un equipo de calidad que tiene que aprobar la calidad del producto final, si no es aceptado vuelve a pasar por los desarrolladores. La toma de decisiones se realiza mediante discusiones o votaciones. La diferencia reside en que el comité de OpenOffice está compuesto por miembros escogidos por la comunidad, mientras que en NetBeans un miembro del comité y parte de los desarrolladores proceden de Oracle.

En este caso tal y como se puede observar en la Tabla 8 predomina el tipo de liderazgo de organización o dictador benevolente, lo que no implica que en otras comunidades utilicen otro tipo de liderazgo. En el caso de haber realizado un estudio más amplio

sobre más comunidades se podrían haber encontrado otros tipos como la comunidad sin organización formal o la organización predominante.

5.2.2 Resultados de las Prácticas de Trabajo

Las prácticas de trabajo son otra dimensión de gobernabilidad según [Capra et al., 2008]. En nuestro estudio hemos encontrado que las comunidades OpenOffice, NetBeans y Eclipse utilizan "Infraestructuras online", solo Mozilla pertenece a la categoría de "Infraestructuras online + Reuniones físicas".

En la Tabla 9 se muestra que las prácticas de trabajo habituales en las comunidades bajo estudio son las infraestructuras online. Esto se debe principalmente a que los miembros de la comunidad se encuentran geográficamente dispersos, por lo que el medio más conveniente para comunicarse son las infraestructuras online. Es cierto que en estas comunidades también se organizan eventos y actividades donde se pueden reunir los participantes, pero solo en Mozilla disponen de oficinas donde se reúnen para desarrollar sus productos. Al igual que antes, el hecho de no haber encontrado otro tipo de prácticas de trabajo no quiere decir que no se apliquen en otras comunidades OSS.

Comunidad	Prácticas	Observaciones
OpenOffice	Infraestructuras online	Los participantes de la comunidad se encuentran geográficamente dispersos, por lo que utilizan infraestructuras online para comunicarse.
Mozilla	Infraestructuras online + Reuniones físicas	Los participantes de la comunidad se encuentran geográficamente dispersos, por lo que utilizan infraestructuras online para comunicarse. Pero disponen también de varias oficinas donde se pueden reunir los colaboradores.
NetBeans	Infraestructuras online	Los participantes de la comunidad se encuentran geográficamente dispersos, por lo que utilizan infraestructuras online para comunicarse.
Eclipse	Infraestructuras online	Los participantes de la comunidad se encuentran geográficamente dispersos, por lo que utilizan infraestructuras online para comunicarse.

Tabla 9. Resultados de las prácticas de trabajo

5.2.3 Resultados de las Características

Después del estudio realizado sobre las cuatro comunidades OSS hemos obtenido datos de las diferentes características relacionadas con el proceso de requisitos. Destacamos las principales fuentes de requisitos, las infraestructuras utilizadas, quienes participan en ellas y quienes están involucrados en la toma de decisiones. Además de si se utiliza alguna técnica del proceso de la Ingeniería de Requisitos, de modo que podemos comparar el proceso de requisitos tradicional con el realizado en las comunidades OSS. En la Tabla 10 se muestran los resultados.

Características	OpenOffice	Mozilla	NetBeans	Eclipse
Principales fuentes de requisitos	Usuarios finales Desarrolladores	Usuarios finales Desarrolladores	Usuarios finales Desarrolladores	Usuarios finales Desarrolladores
Principales infraestructuras de requisitos	Listas de correo Foro Wiki Bugzilla Archivos de correos	Listas de correo Foro Wiki Bugzilla Chat Grupo de noticias	Listas de correo Foro Wiki Bugzilla Chat	Listas de correo Foro Wiki Bugzilla
Participación en las infraestructuras	Usuarios finales Desarrolladores Contribuidores	Usuarios finales Desarrolladores Contribuidores Administrador o propietario del proyecto	Usuarios finales Desarrolladores Contribuidores	Usuarios finales Desarrolladores Contribuidores Administrador o propietario del proyecto
Toma de decisiones	Usuarios desarrolladores y contribuidores pueden estar implicados en todas las fases del proceso	Usuarios desarrolladores y contribuidores pueden estar implicados en todas las fases del proceso	Usuarios desarrolladores y contribuidores pueden estar implicados en todas las fases del proceso	Usuarios desarrolladores y contribuidores pueden estar implicados en todas las fases del proceso
Técnicas del proceso de la Ingeniería de Requisitos	*Brainstorming* en la educción de requisitos	Ninguna	Ninguna	Prototipo y encuesta en la negociación de requisitos

Tabla 10. Resultados de las características

En todas las comunidades observamos características comunes, pero no podemos generalizar estos datos puesto que los casos de estudio se limitan a comunidades OSS muy populares y con un gran número de participantes. Cabe destacar que tanto usuarios como desarrolladores y colaboradores participan activamente en el proceso de requisitos, por medio de las infraestructuras proporcionadas por la comunidad.

En OpenOffice observamos el uso de brainstorming en la educción de requisitos para la versión 4.0.0 de OpenOffice [OO Brainstorming], la infraestructura utilizada es Google Moderator donde además de publicar ideas y comentarios es posible votar a favor o en contra. Los datos son después analizados por voluntarios y administradores, y posteriormente son publicados. En Eclipse encontramos un prototipo sobre la nueva interfaz gráfica de la página web junto con una encuesta para conocer la opinión de los usuarios. Se tratan de casos que afectan a todos los usuarios e interesa su opinión, pero no es muy habitual encontrar este tipo de técnicas en las comunidades OSS, puesto que disponen de otras infraestructuras para educir y analizar los requisitos.

5.2.4 Resultados de la Encuesta a los Usuarios

Para comprobar la participación de la comunidad y como percibían los usuarios el proceso de requisitos realizamos una encuesta a 22 usuarios. Las preguntas de la encuesta se pueden encontrar en el Anexo C. A continuación, analizamos los resultados obtenidos.

En la Figura 1 se muestran las infraestructuras y tecnologías OSS utilizadas por los usuarios. Aunque hay una gran cantidad de herramientas OSS dedicadas al desarrollo de software, se observa que las más utilizadas, Mozilla Firefox y OpenOffice, son las que están más orientadas a usuarios finales.

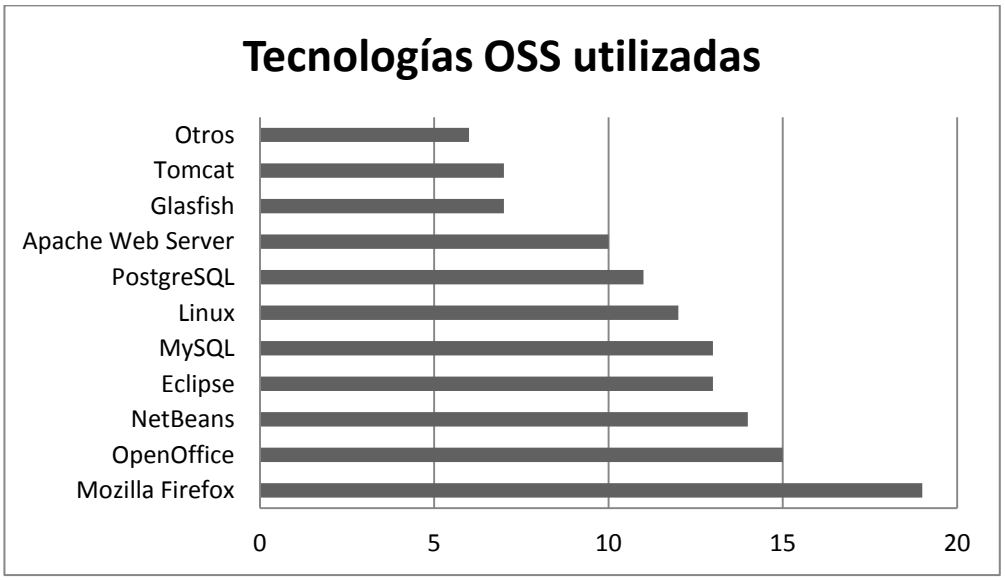


Figura 1. Tecnologías OSS utilizadas por usuarios

La gran mayoría de usuarios reconoce que no interactúa con la comunidad, pero si lo hace, participa en los foros y en menor medida en las listas de correos o wikis. Solo un 20% de los encuestados realiza peticiones de nuevas funcionalidades o informa de errores. Un 85% desconoce cuál es el proceso que siguen las solicitudes y no participan en el desarrollo de las nuevas funcionalidades.

A la hora de resolver dudas o errores más de un 50% de usuarios consulta foros en Internet o consulta a otras personas, sino pasa a consultar libros y manuales de referencia o utiliza la ayuda en línea. Esto nos indica que los usuarios confían en las respuestas dadas por otras personas, siempre que tengan cierta calidad. Lo que nos hace cuestionarnos la eficiencia de las búsquedas en los foros y la valoración de las respuestas. También debemos considerar la accesibilidad a la documentación y los FAQs para aprovechar mejor estos recursos.

5.2.5 Verificación de las Afirmaciones

Seguidamente procederemos a verificar las afirmaciones que realizamos en el Capítulo 4 sobre cada una de las actividades del proceso de requisitos de la Ingeniería de Requisitos.

Educción de Requisitos

En relación con la educción de requisitos, confirmamos que además de usuarios finales, también se encuentran desarrolladores en el aporte de nuevas funcionalidades, y los desarrolladores son a su vez usuarios del software. Los requisitos se plantean de manera informal por lo que en ocasiones los desarrolladores solicitan más detalles. Dependiendo de la comunidad la toma de decisiones difiere, pero las decisiones son discutidas por la comunidad y los propietarios de los módulos, empleando para ello las infraestructuras disponibles.

Análisis de Requisitos

Respecto al análisis de requisitos, se cumple que en las comunidades OSS por lo general los desarrolladores o los jefes de proyecto se encargan de aceptar las peticiones de los usuarios y se encargan de añadirlas a las nuevas versiones del software. En el análisis de los requisitos de la comunidad contribuyen un amplio número de desarrolladores, que son a la vez los usuarios del sistema software. Especialmente se forman discusiones por

medio de las listas de correos, pero desconocemos si se realizan análisis por chat no documentados o diseños individuales por parte de los desarrolladores que no se hacen públicos. De todos modos el código es abierto por lo que se puede realizar un análisis del mismo directamente.

Especificación de Requisitos

Respecto a la especificación de requisitos, uno de los aspectos analizados en este estudio ha sido determinar si las comunidades escogidas mantenían alguna forma de especificar los requisitos en un documento formal. En este caso, en ninguna de las comunidades OSS investigadas en este proyecto se ha encontrado información sobre la publicación de documentación formal de los requisitos funcionales y no funcionales que tiene el sistema de cada comunidad. Sin embargo, tal y como hemos comentado en el transcurso de este estudio, las comunidades OSS utilizan otros tipos de procesos o prácticas online para tratar el tema de la gestión de requisitos, como por ejemplo el uso de infraestructuras como son los foros, las wikis, listas de correo o chats. Es habitual que se publiquen las actualizaciones en las wikis para informar a todos los usuarios que participan en el desarrollo de la comunidad. Por ejemplo en Eclipse tienen un formato específico para las propuestas de proyectos.

Gestión de Requisitos

En relación con la gestión de requisitos, en ninguna de las comunidades se observa ninguna práctica de Ingeniería de Requisitos a la hora de realizar la gestión de los requisitos comparado con la Ingeniería de Requisitos tradicional, sino que tal y como hemos podido observar a lo largo de la investigación, la gestión de los requisitos la llevan a cabo a través del conjunto de infraestructuras que pone a disposición cada comunidad, en su mayoría Bugzilla. Cada comunidad estudiada utiliza sus técnicas, prácticas de trabajo y tipos de liderazgo para la gestión de requisitos y el desarrollo del sistema de la comunidad, lo que no significa que sea una forma mejor o peor de gestionar la Ingeniería de Requisitos en las comunidades OSS. Es decir, cada una de ellas utiliza las prácticas de trabajo que más le convienen para llevar a cabo la Ingeniería de Requisitos. De este modo, es interesante explorar estas prácticas con el fin de enriquecer y/o mejorar los mecanismos tradicionales sugeridos por la Ingeniería de Requisitos tradicional.

Capítulo 6

PARTICIPACIÓN EN EL PROCESO DE REQUISITOS DE PROYECTOS OPEN SOURCE SOFTWARE

Para completar el trabajo desde el punto de vista de un colaborador, en este capítulo detallaremos cómo se han llevado a cabo aportaciones sobre otras comunidades. Incluiremos también la experiencia personal de un desarrollador, de modo que podamos conocer otros aspectos que no se han podido observar en la investigación anterior e identificar diferencias y similitudes entre los procesos de requisitos de comunidades grandes y de proyectos pequeños.

A continuación, se describen las características de las comunidades seleccionadas y el proceso seguido para realizar una aportación, además de una comparación con las comunidades estudiadas en el capítulo anterior. En el Anexo D se describen las herramientas utilizadas.

6.1 Caso de Estudio: Double Commander

DoubleCommander es un administrador de archivos de código abierto, multiplataforma, inspirado en Total Commander [Double Commander, 2014]. Consta de una doble pantalla que permite mover archivos entre directorios de una forma sencilla. Incluye también otras funcionalidades como un editor de texto integrado y un visualizador de archivos en binario, hexadecimal y en formato texto.

Criterios de selección: Este proyecto fue escogido en contraste con las comunidades anteriores, ya que se trata de una comunidad más pequeña, de decenas de usuarios.

Búsqueda: Se encontró en SourceForge. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: El software se gestiona y puede descargarse de forma gratuita desde SourceForge [Dc Descarga].

Contacto: Este proyecto se gestiona principalmente por los servicios que proporciona SourceForge, la mayor parte de los recursos de la página principal redireccionan a SourceForge [Dc Contacto].

En la Tabla 11 se muestran las características de la comunidad Double Commander.

Características
Roles de la comunidad
No se definen de manera formal los roles en la comunidad, pero podemos clasificar los siguientes: - **Usuarios:** son las personas que usan el software, informan de errores y hacen peticiones o sugerencias. - **Colaboradores:** son los miembros que contribuyen pero que no tienen acceso de escritura al código fuente principal, pueden realizar aportaciones de parches, código nuevo o informes de errores, también pueden añadir contenidos como artículos, FAQs o pantallazos. - **Administradores:** son los encargados de juntar los parches de los contribuyentes, las correcciones de errores y el nuevo código, asegurándose de que estas aportaciones son estables y cumplen con la calidad adecuada.
Gobierno de la comunidad
Comunidad sin organización formal: no existe una organización formal, las decisiones son tomadas por los administradores, después de discusiones informales en las que pueden participar usuarios y desarrolladores.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 9 usuarios y 13 contribuidores que se encuentran distribuidos por Europa y Estados Unidos entre otros países. Pero podemos estimar que hay muchos más usuarios por el número de descargas de la aplicación que ronda las 3.000 descargas.
Documentación
Desde la página principal se puede acceder a una guía para desarrolladores en la wiki [Dc Documentación]. No dispone de mucha documentación para usuarios.

Tabla 11. Características de la comunidad Double Commander

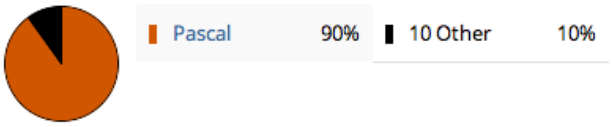
Características
Infraestructuras
Las infraestructuras que utiliza Double Commander se basan en los servicios proporcionados por SourceForge: - **Double Commander Wiki:** indica donde se puede descargar el software y las versiones anteriores, dispone solo de una guía para desarrolladores y hace referencia al foro, disponible en inglés y ruso [Dc Wiki]. - **Mantis Bug Tracker:** un lugar para informar o buscar errores y mejoras [Dc Mantis]. - **Double Commander Forum:** está separado en tres categorías a saber general, sugerencias y discusión de errores [Dc Foro]. - **Listas de Correos:** solo dispone de una lista de correos (doublecmd-devel) para desarrolladores [Dc Listas de correos].
Licencias
GNU General Public License (GPL) v2.0 GNU Library or "Lesser"GPL (LGPL) v2.0
Lenguajes utilizados

Otros
No dispone de página de donaciones, lo que algunos usuarios sugieren añadir. Se observa en los foros que no se responden a todas las sugerencias, ni siquiera negando la implementación o dando una solución alternativa. Pero los contribuidores son activos puesto que sale una actualización cada 2 ó 3 meses.

Tabla 11. Características de la comunidad Double Commander (Continuación)

Después de estudiar las características de la comunidad y de descargar e instalar la aplicación se procedió a conocer su funcionalidad para poder buscar algún nuevo requisito que se pudiese sugerir. Finalmente se dio con una posible mejora, concretamente cuando se exploran los archivos, hay un botón que deja de mostrar el explorador principal para mostrar otra información. Cuando se desea volver a navegar por los archivos, no se puede volver al sitio exacto donde se había dejado.

En la página web del proyecto [Double Commander, 2014] se encuentra toda la información necesaria para poder reportar incidencias, realizar propuestas y así poder contribuir a la mejora de la aplicación. Mantis es el medio utilizado para proponer nuevas funcionalidades y reportar errores. El error o aportación es recibido por los desarrolladores, quienes confirman el error y lo asignan a los miembros de la comunidad que pidan trabajar voluntariamente en dicha tarea.

Con el objetivo de ver la reacción a la propuesta y el seguimiento de la misma, se inició un contacto con la comunidad para proponer implementar un nuevo botón que, tras

acceder a la información, permitiese volver a la misma ruta por la que estabas navegando. Tras hacer la propuesta, se recibió una contestación de la comunidad en la que se denegaba el desarrollo, ya que existía un comando que implementaba la funcionalidad propuesta. Actualmente se puede volver al directorio utilizando el historial, ya sea haciendo click en el botón historial y buscando la ruta deseada o usando un comando, pero no se trataba exactamente de lo que se proponía.

La propuesta realizada buscaba sencillez y rapidez mediante un botón volver. El motivo de no aprobarla posiblemente se encuentre en que este tipo de aplicación no está orientada a usuarios sin ningún conocimiento técnico, por lo que el uso del comando implica directamente las características propuestas. El perfil de usuario que usa esta aplicación es un usuario con cierto conocimiento técnico por lo que el hecho de que las cosas sean *user-friendly* no es muy valorado.

La persona que denegó el aporte fue un desarrollador, que contaba con derechos de administrador en Mantis. Los usuarios pueden añadir sus comentarios, pero la potestad sobre qué se desarrolla y qué no la tienen los administradores de la aplicación.

6.2 Caso de Estudio: FileZilla

FileZilla es un cliente FTP, FTPS y SFTP que permite cargar o descargar archivos en o desde un servidor [FileZilla, 2014]. Es una herramienta muy útil para desarrolladores de software ya que, por ejemplo, permite cargar los componentes de un sitio web en su servidor, o descargarlos desde remoto. FileZilla tiene una interfaz de usuario gráfica e intuitiva y es un sistema rápido y confiable.

Criterios de selección: Este proyecto fue escogido en contraste con las comunidades anteriores, ya que se trata de una comunidad más pequeña pero a su vez activa.

Búsqueda: Se encontró en SourceForge como recomendada por los editores. Para obtener más información se observaron los datos analizados en Ohloh.net y se realizó una búsqueda más profunda en su página web oficial, accediendo para el estudio a las secciones de discusión para desarrolladores, como los foros, wikis y listas de correo.

Software: El software cliente y servidor se pueden descargar de forma gratuita desde SourceForge o desde la página de descargas de la comunidad [Fz Descarga].

Contacto: En su página web, existe una sección en la que se informa sobre las formas en las que se puede participar [Fz Contacto].

En la Tabla 12 se muestran las características de la comunidad FileZilla.

Características
Roles de la comunidad
No se definen de manera formal los roles en la comunidad, pero podemos clasificar los siguientes: - **Usuarios:** son las personas que usan el software, informan de errores y hacen peticiones o sugerencias. - **Colaboradores:** son los miembros que contribuyen pero que no tienen acceso de escritura al código fuente principal, pueden realizar aportaciones de parches, código nuevo o informes de errores, también pueden añadir contenidos como artículos, FAQs o pantallazos. - **Administradores:** son los encargados de juntar los parches de los contribuyentes, las correcciones de errores y el nuevo código, asegurándose de que estas aportaciones son estables y cumplen con la calidad adecuada. Cada módulo tiene un encargado que tiene permisos de registro y gestiona a un grupo de desarrolladores.
Gobierno de la comunidad
Comunidad sin organización formal: no existe una organización formal, las decisiones son tomadas por los administradores, después de discusiones informales en las que pueden participar usuarios y desarrolladores.
Distribución de la comunidad
Según Ohloh.net está compuesta en total de 1.968 usuarios y 13 contribuidores que se encuentran distribuidos por Europa y América entre otros países.
Documentación
La wiki dispone de documentación y FAQs clasificadas para el cliente y el servidor de FileZilla, además de una guía para compilar el código para los desarrolladores [Fz Documentación].
Infraestructuras
- **FileZilla Wiki:** indica dónde se puede descargar el software, recoge la documentación y las FAQs e indica cómo contactar con la comunidad [Fz Wiki]. - **FileZilla's Trac:** un sistema de tickets para informar o buscar errores y mejoras [Fz Trac]. - **FileZilla Forums:** está separado en distintos temas donde los administradores y moderadores trabajan activamente [Fz Foro]. - **FileZilla Chat:** se trata de un chat IRC soportado por freenode [Fz Chat].
Licencias
GNU Library or "Lesser" GPL (LGPL) v2.0

Tabla 12. Características de la comunidad FileZilla

Características
Lenguajes utilizados
 C++ 63% C 27% XML 5% 8 Other 5%
Otros
En ocasiones los administradores rechazan de forma tajante las nuevas propuestas.

Tabla12. Características de la comunidad FileZilla (Continuación)

En la página web de FileZilla Project se puede encontrar toda la funcionalidad necesaria para realizar los distintos tipos de colaboraciones con la comunidad. Además contiene una serie de manuales donde se describe cómo realizar la comunicación con la comunidad, describiendo tanto los medios como las formas. Por ejemplo, se puede encontrar un manual de cómo realizar una petición de cambio o reportar un error y un manual de cómo redactar correctamente un comentario contestando a otro usuario.

En este caso tras registrarse en su comunidad, se intentó colaborar como usuario de todas las formas posibles, eligiendo en primer lugar añadir un posible cambio en los atajos por teclado, puesto que según el entorno un mismo comando realizaba acciones distintas. El problema consistía en que cuando se estaba trabajando con archivos locales, si se desea editarlos de forma directa, se puede navegar hasta ellos y pulsando la tecla E iniciar la edición, apareciendo un *popup* de confirmación, que se acepta pulsando ENTER, por lo que la pulsación de E + ENTER se hace de forma casi inmediata. Si por el contrario se está trabajando en remoto, por ejemplo viendo los ficheros de un servidor, al pulsar E se eliminan archivos, abriéndose una ventana de confirmación. Al estar acostumbrado a editar, la primera vez que se pulsa E + ENTER sin prestar atención se eliminan archivos.

Para ello se realizó una petición dentro de la sección de errores. Pasado un tiempo se recibe una comunicación por correo donde se informa que la petición no está en el sitio correcto y que se ha creado en la zona de futuros cambios con una prioridad alta, adjuntando el id de la petición, lo que permite llevar un seguimiento de las aportaciones de los usuarios. Y al poco tiempo se añade una respuesta donde se asegura haber solucionado el problema, los administradores cambian el tipo a la petición, pasándola a *patch*, siendo una mejora para las próximas versiones.

Durante este periodo, se observa cómo en la comunidad se da gran importancia a las traducciones, llevando un seguimiento exhaustivo del estado de las mismas según el

idioma. Como el español no estaba finalizado, se procede a contribuir en la traducción, siguiendo el *workflow* que se describía en la página.

La siguiente aportación consiste en colaborar con la traducción de la aplicación. La página del proyecto contiene una sección donde se pueden ver todas las traducciones que se han realizado y el estado en el que se encuentran. También contiene los pasos a seguir para realizar una traducción, permitiendo de forma *online* compilar los ficheros para poder probarlos previamente en la aplicación, en caso de tener descargado el código. Para completar el castellano se descarga el fichero correspondiente y se procede a realizar la traducción. Una vez implementada se envía la traducción a un correo que se facilita en la página, donde comprueban que es correcto y actualizan la información.

6.3 Caso de Estudio: Social Network Manager

En esta ocasión se estudia la colaboración en un desarrollo para un futuro proyecto OSS dedicado a la gestión de redes sociales, Social Network Manager. La colaboración se obtuvo desde la página openhatch.org, en la que se ofertan colaboraciones en proyectos OSS, y donde es posible realizar búsquedas de proyectos a los que vincularse mediante los lenguajes de programación. La colaboración seleccionada consistía en la implementación de una funcionalidad para medir relaciones entre usuarios de Twitter, el criterio de selección fue que el lenguaje en el que se debía realizar el desarrollo era bien conocido por el desarrollador.

En la descripción de la aportación se incluía una dirección de correo a la que se escribió para ofrecer la colaboración, puesto que todavía no se ha terminado de desarrollar el código no se dispone de otras infraestructuras que den soporte al software. Al poco tiempo se recibió una contestación afirmativa y se iniciaron una serie de comunicaciones para la realización del desarrollo.

Para ello enviaron por correo una descripción de la funcionalidad deseada, de la que se podía obtener una lista de requisitos. Se trataba de una simple descripción, de cómo debía responder la aplicación, no seguía ningún tipo de estándar ni contenía un listado con los requisitos separados según el tipo, era un texto en el que se especificaba de forma detallada el módulo a implementar. Esa información fue suficiente para comprender qué se pedía. Aparte de esta descripción, se recibió una serie de funciones PHP en un archivo comprimido que permitían usar la base de datos (BBDD) de la

aplicación, pero no se tenía acceso a él, solo se disponía de sus descripciones bien documentadas y organizadas. Por último se recibió por mail un manual de uso de la Api de Twitter, en el que se explicaba el funcionamiento de las diferentes llamadas que podían resultar útiles.

La primera petición que se hizo al desarrollador fue la de llevar a cabo un diseño de cómo se pretendía realizar la implementación, para poder evaluar el rendimiento que tendría. Tras pensar en la mejor solución posible se redactó un documento en el que se explicaba el diseño propuesto, indicando las funciones utilizadas y el flujo que se seguiría. En poco tiempo se recibió una contestación en la que se felicitaba la solución propuesta y se daba una serie de recomendaciones para desarrollar el código, organizarlo y probarlo.

No se pidió seguir ningún estándar para la implementación, simplemente sentido común teniendo en cuenta que ese código debía ser legible para otros usuarios. Una vez desarrollado el código, se probó teniendo en cuenta las recomendaciones recibidas de la comunidad. Tras varias pruebas y ajustes se envió a la comunidad. Al cabo de unas semanas contestaron explicando que funcionaba correctamente y que estaban muy agradecidos. De hecho preguntaron por la disponibilidad del desarrollador para continuar con la participación en un futuro.

6.4 Comparación de Comunidades OSS

En todas las comunidades estudiadas el proceso de requisitos es muy parecido, en todos ellos son los desarrolladores y los usuarios finales los que aportan ideas o reportan errores, por medio de diferentes infraestructuras online, foros, listas de correo u otros sistemas. Habitualmente se discute la necesidad del nuevo requisito y es un desarrollador o un administrador el que decide aplicar la mejora, en este punto las comunidades pueden diferir según el tipo de gobierno que apliquen. La educción, análisis y especificación de requisitos no está formalizado en ninguna de las comunidades estudiadas.

Es posible afirmar que cualquier usuario común puede aportar nuevas funcionalidades a una comunidad, pudiendo buscar proyectos dentro de sus intereses. También queda patente que las infraestructuras disponibles permiten reportar errores o nuevas funcionalidades de forma directa y sencilla, solo se requiere del registro con un usuario

y una contraseña. Aunque se puede apreciar que en las comunidades grandes disponen de más recursos para los usuarios, más documentación, guías para usuarios y tutoriales y el acceso en distintos idiomas. Una diferencia notable en las comunidades organizadas es la actividad en los foros y en las listas de correos, que escasean en las comunidades poco numerosas.

En los últimos casos de estudio podemos apreciar que se aprovechan como soporte las infraestructuras que proporciona SourceForge, mientras que las otras comunidades estudiadas disponen de sus propias infraestructuras. Al igual que en SourceForge o la mencionada OpenHatch, podemos encontrar de forma online muchos proyectos en desarrollo en los que se pueden realizar contribuciones. Observando la aplicación de gestión de proyectos hay que destacar el elevado número de participantes y la gran cantidad de mejoras en las que se puede trabajar, del orden de miles.

Capítulo 7

CONCLUSIONES Y TRABAJO FUTURO

En este último capítulo comentaremos las conclusiones alcanzadas tras los análisis sobre el estudio realizado y los resultados obtenidos en los capítulos anteriores, con el fin de determinar y comprender las actividades llevadas a cabo en el proceso de requisitos en comunidades OSS. Además se incluirán posibles trabajos futuros que continúen con esta línea de investigación y contribuyan al entendimiento y desarrollo de las comunidades OSS.

Algunos estudios han demostrado que los procesos OSS son diferentes en muchos aspectos de los procesos software tradicionales [Scacchi et al., 2005][Tian, 2006][Potdar y Chang, 2004]. Pero algunos autores como Fuggetta [Fuggetta, 2003] y Godfrey y Tu [Godfrey y Tu, 2000] no son de tal opinión y afirman que el modelo de desarrollo OSS no es exactamente una nueva descripción del proceso, sino sólo una visión alternativa de las actividades de Ingeniería del Software aplicadas a los modelos de desarrollo tradicional. El trabajo realizado apunta a determinar si realmente son diferentes, enfocándose en el proceso de requisitos. Después de concluir el trabajo podemos decir que los objetivos de los procesos tradicionales y OSS son los mismos, pero difieren completamente en las técnicas, herramientas e infraestructuras para alcanzar estos objetivos, tal y como indican Scacchi, Tian, Potdar y Chang. Salvo un par de excepciones donde hemos encontrado técnicas tradicionales como *brainstorming* o prototipos y encuestas en comunidades OSS.

El proceso de requisitos es un conjunto de actividades relevante en cualquier proyecto de software. Con el objetivo de comprender dicho proceso en las comunidades OSS, hemos realizado un estudio sobre varias comunidades con éxito, OpenOffice, Mozilla, NetBeans y Eclipse. En contraste con otras comunidades no tan populares, Filezilla y Double Commander en las que se han realizado aportes y su seguimiento.

Tras el análisis de los resultados hemos observado que las comunidades tienen ciertas características comunes. La principal fuente de requisitos proviene de la necesidad de desarrolladores y usuarios finales, pero la toma de decisiones final suele estar en manos de un comité escogido por la comunidad por meritocracia, o por un jefe de proyecto que voluntariamente se hace responsable, lo que llamamos "Organización o dictador benevolente". Aun así los desarrolladores y usuarios pueden participar activamente en las discusiones, por medio de las infraestructuras disponibles ya sean listas de correos, foros o chats. Debido a que la mayoría de los miembros de las comunidades se encuentran dispersos, la comunicación y la organización se realiza necesariamente a través de infraestructuras online. Aunque también se pueden realizar periódicamente reuniones físicas. Especialmente hemos observado en las comunidades exitosas una sección para el marketing y la formación sobre el uso del software desarrollado lo que contribuye a ampliar la participación en las comunidades mediante eventos.

Después del trabajo realizado no hemos encontrado documentación formal sobre la especificación de requisitos, solo quedan documentados los requisitos en las infraestructuras donde se discuten de manera informal y se deja su implementación a la interpretación del desarrollador. Esto es un inconveniente puesto que los desarrolladores son voluntarios y en ocasiones abandonan el proyecto, de manera que el próximo contribuyente que retome el proyecto no dispone de documentación sobre el avance del mismo, lo que no ocurre siguiendo el proceso tradicional. La infraestructura habitualmente utilizada por las comunidades para realizar las gestiones de requisitos es un sistema de tickets, el más utilizado en las comunidades estudiadas es Bugzilla, aunque encontramos otros como Mantis.

Como conclusión, cada comunidad tiene su propia organización y aprovecha las infraestructuras que tiene a su disposición para realizar de la forma que considera más apropiada el proceso de requisitos. El conocimiento sobre los procesos que utilizan las comunidades OSS puede llegar a aportar mejoras en los procesos tradicionales y viceversa beneficiándose mutuamente. Por ejemplo, el modo en el que las comunidades educen requisitos o el seguimiento que pueden hacer los usuarios de su desarrollo en infraestructuras online se puede aplicar en la industria del software, tanto en el proceso de desarrollo como en el mantenimiento. Por parte del OSS deberían adoptar un conjunto de criterios más formalizado para la documentación, de modo que sea más

accesible para los nuevos colaboradores conocer los requisitos propuestos y el trabajo realizado.

En el ámbito personal este trabajo me ha permitido completar mi formación adquiriendo conocimientos sobre el entorno de OSS poco conocido, puesto que habitualmente se estudia el procedimiento tradicional de la Ingeniería del Software. Además me ha sorprendido gratamente la cantidad de participantes que colaboran de forma voluntaria en proyectos OSS y, de esta manera, este trabajo realizado me ha facilitado la adquisición de competencias para el mundo laboral.

A continuación se plantean una serie de posibles trabajos futuros:

- Replicar este trabajo con más casos de estudio, para ampliar así los conocimientos sobre el proceso de requisitos en otras comunidades OSS.
- Realizar estudios sobre las comunidades OSS con diferentes características profundizando en otras etapas de la Ingeniería del Software, como por ejemplo el proceso de implementación y pruebas junto con el control de calidad y de configuraciones, con el fin de obtener más información sobre cómo las comunidades realizan la gestión del software que desarrollan.
- Estudiar nuevas infraestructuras que se puedan adaptar a las comunidades para mejorar su comunicación, tales como las redes sociales u otras herramientas para hacer más eficiente la documentación sobre el desarrollo del software, ya que actualmente se utilizan las listas de correo o los foros y si no están organizados adecuadamente se trata de un inconveniente para nuevos usuarios inexpertos. ¿Sería posible mejorar el sistema de búsqueda o aplicar un etiquetado en las infraestructuras para facilitar las búsquedas? Además en estas infraestructuras el hilo de mensajes que se forma no es cómodo de leer y los mensajes más antiguos se quedan en el olvido.
- Una observación que hemos podido apreciar en este trabajo es que gran parte de estas comunidades tenían un equipo de marketing que publicitaba y fomentaba el uso y formación sobre el software, lo que puede ser causa de su éxito y de la supervivencia del OSS. Puede que no tenga relación directa con la materia que nos incumbe, pero se considera un futuro caso de estudio, el proceso de marketing o el proceso de formación como factor importante en la colaboración y el desarrollo del software.

Referencias

Acuña, S.T., Castro, J.W., Dieste, O., Juristo, N. (2012). A Systematic Mapping Study on the Open Source Software Development Process. *Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering* (EASE'12), Ciudad Real, Spain, pp. 42-46.

Bray, I. (2002). An Introduction to Requirements Engineering. Addison-Wesley, Pearson Education, Harlow, UK, pp. 23-223.

Capra, E., Francalanci, C., Merlo, F. (2008). An Empirical Study on the Relationship among Software Design Quality, Development Effort an Governance in Open Source Projects. *IEEE Transactions on Software Engineering*, vol. 34 nº 6, pp. 765-782 November / December, First International Workshop on Emerging Trends in FLOSS Research and Development. IEEE Computer Society.

Castro, J.W., Acuña, S.T. (2012). Differences between Traditional and Open Source Development Activities. *Proceedings of the 13th International Conference on Product-Focused Software Development and Process Involvement* (PROFES'12), Madrid, Spain, pp. 131-144.

Escribano, Y. (2011). Estado de la Práctica de la Ingeniería de Requisitos en Proyectos de Software Open Source. Universitat Politècnica de Catalunya Departament de Llenguatges i Sistemes Informàtics, Barcelona, España, pp. 280.

Fuggetta, A. (2003). Open Source Software: An Evaluation. *Journal of System and Software*, vol. 66, pp. 77-90.

Gacek, C., Arief, B. (2004). The Many Meanings of Open Source. University of Newcastle upon Tyne. *IEEE Software*, vol. 21 nº 1, pp. 34-40.

Gerea, M. (2007). Selection of Open Source Components – A Qualitative Survey in Norwegian IT Industry. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Godfrey, M.V., Tu, Q. (2000). Evolution in Open Source Software: A Case Study. Proceedings of the *International Conference Software Maintenance* (ICSM'00). San José, CA, pp. 131-142.

Hauge, O., Sorensen, C., Rosdal, A. (2007). Surveying Industrial Roles in Open Source Software Development. Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

Heliö, J. (2004). Requirements Engineering when Collaborating with Open Source Projects: Digital Business Ecosystem. Tampere University of Technology, Tampere, Finland.

INCOSE (2000). How to: Guide for all Engineers Version 2. International Council on Systems Engineering. Seattle, WA, USA.

Kotonya, G., Sommerville, I. (2000). Requirements Engineering: Processes and Techniques. Jhon Wiley & Sons, Chichester, UK, pp. 294.

Laurent, P., Cleland-Huang, J. (2009). Lessons Learned from Open Source Projects for Facilitating Online Requirements Processes. Systems and Requirements Engineering Center, School of Computing, DePaul University, Chicago, IL, USA.

Mockus, A., Fielding, R.T., Herbsleb, J. (2000). A Case Study of Open Source Software Development: The Apache Server. *Proceedings of the 22th International Conference on Software Engineering* (ICSE'00), Limerick, Ireland, pp. 243-272.

Mockus, A., Fielding, R.T., Herbsleb, J. (2002). Two Case Studies of Open Source Software Development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology*, vol. 11 nº 3, pp. 309-346.

Potdar, V., Chang, E. (2004). Open Source and Closed Source Software Development Methodologies. Proceedings of the *26th International Conference on Software Engineering*, Edinburgh, UK, pp. 105-109.

Robson, C. (1994). Experiment, Design and Statistics in Psychology. 3rd edition, Penguin Books, London, England.

Runesson, P., Höst, M. (2009). Guidelines for Conducting and Reporting Case Study Research in Software Engineering. *Empirical Software Engineering*, vol. 14 nº 2, pp. 131-164.

Scacchi, W. (2002). Understanding Requirements for Developing Open Source Software Systems. Institute for Software Research, University of California-Irvine, Irvine, USA.

Scacchi, W. (2004). Free and Open Source Development Practices in the Game Community. *IEEE Software*, vol. 21 nº 1, pp. 59-66.

Scacchi, W., Jesenm, C., Noll, J., Elliot, M. (2005). Multi-Modal Modeling of Open Software Requirements Process. *Proceedings of the First International Conference on Open Source Systems*, Genova, Italy, pp. 1-8.

SWEBOK (2004). Guide to the Software Engineering Body of Knowledge. IEEE Computer Society Professional Practices Committee. Los Alamitos, CA, USA.

Tian, Y. (2006.) Developing an Open Source Software Development Process Model Using Grounded Theory. Universidad of Nebraska. Lincoln, NB, USA, pp. 143.

Enlaces OSS

Ohloh (2014): https://www.ohloh.net/ [Accessed: 11-06-2014]

SourceForge (2014): http://sourceforge.net/ [Accessed: 11-06-2014]

OpenOffice (2014): http://www.openoffice.org/ [Accessed: 12-06-2014]

[OO Descarga] Download Apache OpenOffice (2014): http://www.openoffice.org/download/index.html [Accessed: 13-06-2014]

[OO Contacto] Contact Us OpenOffice (2014): http://www.openoffice.org/contact_us.html [Accessed: 13-06-2014]

[OO Gobierno] Governance (2014): http://www.openoffice.org/about_us/governance.html [Accessed: 18-06-2014]

[OO Documentación] The Apache OpenOffice Documentation Project (2014): https://wiki.openoffice.org/wiki/Documentation [Accessed: 13-06-2014]

[OO Wiki] The Apache OpenOffice Wiki (2014): https://wiki.openoffice.org/wiki/Main_Page [Accessed: 13-06-2014]

[OO Issues] Issue Tracker (2014): http://www.openoffice.org/qa/issue_handling/submission_gateway.html [Accessed: 13-06-2014]

[OO Bugzilla] Apache OpenOffice (AOO) Bugzilla (2014): https://issues.apache.org/ooo/ [Accessed: 13-06-2014]

[OO Foro] Apache OpenOffice Forum (2014): https://forum.openoffice.org/en/forum/ [Accessed: 20-06-2014]

[OO Listas de correos] Apache OpenOffice Public Mailing Lists (2014): https://openoffice.apache.org/mailing-lists.html#users-mailing-list-public [Accessed: 20-06-2014]

[OO Brainstorming] Confluence AOO4 Brainstorming (2014): https://cwiki.apache.org/confluence/display/OOOUSERS/AOO4+Brainstorming [Accessed: 19-06-2014]

[OO PMC FAQs] PMC FAQS (2014): http://openoffice.apache.org/pmc-faqs.html [Accessed: 18-06-2014]

[OO Community] OO Community FAQs (2014): http://openoffice.apache.org/community-faqs.html [Accessed: 18-06-2014]

Mozilla (2014): http://www.mozilla.org/es-ES/ [Accessed: 12-06-2014]

[Mz Descarga] Built for the Web (2014): https://www.mozilla.org/es-ES/products/ [Accessed: 13-06-2014]

[Mz Contacto] Where is Mozilla? (2014): https://www.mozilla.org/en-US/contact/communities/spain/ [Accessed: 27-06-2014]

[Mz Gobierno] Governance (2014): https://www.mozilla.org/en-US/about/governance/ [Accessed: 27-06-2014]

[Mz Documentación] Mozilla Hispano Documentación (2014): https://www.mozilla-hispano.org/documentacion/Portada [Accessed: 27-06-2014]

[Mz Wiki] MozillaWiki (2014): https://wiki.mozilla.org/Main_Page [Accessed: 27-06-2014]

[Mz Bugzilla] Bugzilla@Mozilla (2014): https://bugzilla.mozilla.org/ [Accessed: 27-06-2014]

[Mz Foro] Forums (2014): https://www.mozilla.org/about/forums/ [Accessed: 27-06-2014]

[Mz Chat] MozillaWiki IRC (2014): https://wiki.mozilla.org/IRC [Accessed: 27-06-2014]

[Mz MDN] MDN Mozilla Developer Network (2014): https://developer.mozilla.org/es/ [Accessed: 27-06-2014]

[Mz Proceso] Development process overview (2014): https://developer.mozilla.org/en-US/docs/Mozilla/Developer_guide/Development_process_overview [Accessed: 27-06-2014]

NetBeans (2014): https://netbeans.org/ [Accessed: 12-06-2014]

[Nb Descarga] NetBeans IDE 8.0 Download (2014): https://netbeans.org/downloads/index.html [Accessed: 13-06-2014]

[Nb Contacto] Contact the NetBeans Team (2014): https://netbeans.org/about/contact.html [Accessed: 13-06-2014]

[Nb Gobierno] NetBeans Governance & Project Roles (2014): https://netbeans.org/about/os/governance.html [Accessed: 13-06-2014]

[Nb Contribuidores] NetBeans Community Contributors (2014): https://netbeans.org/community/contributors.html) [Accessed: 23-06-2014]

[Nb Documentación] Documentation, Training & Support (2014): https://netbeans.org/kb/index.html [Accessed: 13-06-2014]

[Nb Wiki] NetBeans Wiki (2014): http://wiki.netbeans.org/Main_Page [Accessed: 23-06-2014]

[Nb Issues] Issues = Bugs, Feature Requests, and Enhacements (2014): https://netbeans.org/community/issues.html [Accessed: 13-06-2014]

[Nb Foro] NetBeans Forums (2014): http://forums.netbeans.org/ [Accessed: 23-06-2014]

[Nb Listas de correos] Top-Level Mailing Lists (2014): https://netbeans.org/community/lists/top.html [Accessed: 23-06-2014]

[Nb Chat] FaqChat (2014): http://wiki.netbeans.org/FaqChat [Accessed 23-06-2014]

[Nb Proceso] IssueLifeCycle (2014): http://wiki.netbeans.org/IssueLifeCycle [Accessed: 25-06-2014]

Eclipse (2014): https://www.eclipse.org/home/index.php [Accessed: 12-06-2014]

[Ec Descarga] Eclipse Downloads (2014): https://www.eclipse.org/downloads/ [Accessed: 13-06-2014]

[Ec Contacto] Contact Us (2014): https://www.eclipse.org/org/foundation/contact.php [Accessed: 13-06-2014]

[Ec Gobierno] Governance Documents (2014): https://www.eclipse.org/org/documents/ [Accessed: 13-06-2014]

[Ec Directores] Eclipse Foundation Board of Directors (2014): https://www.eclipse.org/org/foundation/directors.php [Accessed: 13-06-2014]

[Ec Documentación] Eclipse documentation (2014): https://www.eclipse.org/documentation/ [Accessed: 25-06-2014]

[Ec Wiki] Eclipsepedia (2014): https://wiki.eclipse.org/index.php/Main_Page [Accessed: 25-06-2014]

[Ec Bugzilla] Eclipse Bugzilla (2014): https://bugs.eclipse.org/bugs/ [Accessed: 25-06-2014]

[Ec Foro] Eclipse Community Forums (2014): https://www.eclipse.org/forums/ [Accessed: 26-06-2014]

[Ec Listas de correos] Eclipse Mailing Lists (2014): https://dev.eclipse.org/mailman/listinfo [Accessed: 25-06-2014]

[Ec Proceso] Eclipse Development Process (2014): https://www.eclipse.org/projects/dev_process/development_process.php [Accessed: 25-06-2014]

[Ec Propuestas] Proposals (2014): https://www.eclipse.org/projects/tools/proposals.php [Accesed: 26-06-2014]

Double Commander (2014): http://doublecmd.sourceforge.net/ [Accessed: 12-07-2014]

[Dc Descarga] Download: http://sourceforge.net/p/doublecmd/wiki/Download/ [Accessed: 12-07-2014]

[Dc Contacto] SourceForge Double Commander (2014): http://sourceforge.net/projects/doublecmd/?source=directory [Accessed: 12-07-2014]

[Dc Documentación] Development (2014): http://doublecmd.sourceforge.net/mediawiki/index.php/Development [Accessed: 12-07-2014]

[Dc Wiki] Double CommanderWiki (2014): http://doublecmd.sourceforge.net/mediawiki/index.php/Main_Page [Accessed: 12-07-2014]

[Dc Mantis] Mantis (2014): http://doublecmd.sourceforge.net/mantisbt/my_view_page.php [Accessed: 12-07-2014]

[Dc Foro] Double Commander Official Forum (2014): http://doublecmd.sourceforge.net/forum/ [Accessed: 12-07-2014]

[Dc Listas de correos] Mailing List for Developers (2014): http://sourceforge.net/p/doublecmd/mailman/ [Accessed: 12-07-2014]

FileZilla (2014): https://filezilla-project.org/index.php [Accessed: 12-07-2014]

[Fz Descarga] Download http://sourceforge.net/projects/filezilla/?source=navbar [Accessed: 12-07-2014]

[Fz Contacto] Contact Information (2014): https://filezilla-project.org/contact.php [Accessed: 12-07-2014]

[Fz Documentación] Documentation (2014): https://wiki.filezilla-project.org/Documentation [Accessed: 12-07-2014]

[Fz Wiki] Filezilla Wiki (2014): https://wiki.filezilla-project.org/Main_Page [Accessed: 12-07-2014]

[Fz Trac] FileZilla's Trac (2014): http://trac.filezilla-project.org/ [Accessed: 12-07-2014]

[Fz Foro] FileZilla Forums (2014): https://forum.filezilla-project.org/ [Accessed: 12-07-2014]

[Fz Chat] Filezilla at freenode (2014): http://freenode.net/irc_servers.shtml [Accessed: 12-07-2014]

Anexos

Anexo A. Bitácoras

En este anexo se recogen las actividades realizadas durante el estudio en forma de bitácoras.

Miércoles 11 de junio de 2014

Comienzo haciendo una vista rápida de los dos grandes servidores de proyectos OSS, Ohloh.net y SourceForge.net. Antes de escoger los proyectos para el estudio, observo que además de ofrecer el código o el enlace a sus fuentes, los dos servidores facilitan la contribución en los distintos proyectos ofreciendo herramientas de soporte.

Tras las sugerencias de mi tutora decido realizar el estudio sobre las siguientes comunidades: OpenOffice, Mozilla, NetBeans y Eclipse. Las razones de mi elección se deben a que ya había utilizado el software de dichas comunidades con anterioridad y no necesitaba demasiado esfuerzo y tiempo en la instalación y en la comprensión del software, a fin de realizar un análisis profundo de dichas comunidades. Además de que se tratan de proyectos OSS con bastante éxito y en el caso de OpenOffice y Mozilla se benefician de una comunidad numerosa. El único problema de esta elección es que restringe los resultados de la investigación a comunidades OSS con este tipo de perfil, necesitaré tener en cuenta en los resultados la posibilidad de que existen otras comunidades con diferentes características.

Jueves 12 de junio de 2014

Tras la elección de los proyectos que formarán parte del estudio, accedí a las páginas web principales de cada una para familiarizarme con su infraestructura y obtener información sobre el software que desarrollan, aunque como ya he mencionado antes ya lo había utilizado con anterioridad y tenía ciertos conocimientos sobre ello.

Accedí a la página web de la comunidad española de OpenOffice pero había varios enlaces que se redirigían a la página oficial en inglés, comparé las mismas páginas y observé que la página original ofrecía más información. Por lo que tomé la decisión de realizar el estudio sobre las páginas originales aunque hubiese versiones en español.

Me familiaricé con las páginas de cada comunidad, la única en la que me resultó más difícil ubicarme fue Mozilla, porque en muchos enlaces se redirigía a la página de colaboración.

Viernes 13 de junio de 2014

Para realizar una descripción más completa sobre las comunidades de cada proyecto realicé una búsqueda más profunda, para obtener características y datos como el gobierno de la comunidad, si existía alguna empresa asociada, las licencias que disponían o la documentación proporcionada, lo que queda reflejado en el apartado 5.1 del Capítulo 5. Comienzo con la comunidad de OpenOffice.

Lunes 16 de junio de 2014

Accedo al foro español de OpenOffice comenzando con la búsqueda de la educción de nuevos requisitos. Tras cierta incertidumbre al comienzo de mi búsqueda, finalmente doy con un tema en el foro donde se realizan aportaciones y no solo consultas o informes de errores como se pueden encontrar en la mayor parte del foro.

Martes 17 de junio de 2014

Después de una reunión con mi tutora decidimos revisar la metodología aplicada en este estudio teniendo en mente el objetivo del mismo, con lo que se reescribieron las preguntas en un principio planteadas, puesto que estas anteriormente se centraban solo en la gestión de requisitos y las infraestructuras utilizadas en las comunidades de OSS.

Miércoles 18 de junio de 2014

Retomé el estudio desde otra perspectiva, no solo dedicarme a buscar nuevos requisitos en foros o listas de correos, sino realizar un seguimiento de los mismos en su desarrollo. Para ello tuve que realizar una búsqueda más profunda sobre dónde y cómo se gestionaban los requisitos. En esta ocasión accedí a la página oficial de OpenOffice y encontré enlaces a la wiki con información sobre los procesos que utiliza la comunidad para tomar decisiones y un repositorio de Apache donde se recopilaban versiones anteriores y se informaba de la gestión de los proyectos.

Jueves 19 de junio de 2014

He vuelto a acceder al repositorio de Apache para realizar una investigación más profunda y he encontrado un enlace donde se realizó una tormenta de ideas para la

versión 4 de OpenOffice y también he podido observar los resultados de votaciones y la gestión de tareas.

Viernes 20 de junio de 2014

Continué con la observación y el análisis de las listas de correo y foros de OpenOffice. En los foros encontré una sección donde se pueden proponer ideas y sugerencias. Al tratarse del foro español, estas sugerencias han de comunicarse después a la página principal, por lo que parece ser que el moderador de la comunidad española es el responsable de establecer esta comunicación. Si ya existe la propuesta, se ofrece un enlace para votar, o si no, se busca alguna alternativa. La mayor parte de sugerencias que encontramos en el foro parecen ser de usuarios, algunas sugerencias se acompañan de imágenes gráficas para la explicación del problema o necesidad.

En las listas de correos la mayoría de los mensajes se tratan de peticiones de ayuda, dudas o errores, según la lista a la que accediéramos éstas podían ser de usuarios o desarrolladores. En ocasiones los usuarios no saben si se trata de un bug o si han hecho algo mal, de todas maneras siempre se intenta dar una respuesta. Las respuestas a estos mensajes normalmente son dadas por desarrolladores y sus explicaciones suelen ser indicaciones paso a paso de cada instrucción o referencias a la documentación existente.

En la parte de desarrolladores podemos encontrar además de dudas y de discusiones, propuestas, anuncios y voluntarios que buscan cómo colaborar. Se comprueban que las sugerencias no están ya implementadas o si existe una anterior que ya esté en desarrollo, sino se pasa la propuesta a Bugzilla para gestionarla. Las discusiones se realizan habitualmente en las listas de correo y después alguien se hace responsable de forma voluntaria de actualizar la información en la wiki.

Lunes 23 de junio de 2014

Después del estudio de OpenOffice pasé a investigar NetBeans, que se trata de la comunidad más pequeña entre las que hemos escogido. En su wiki podemos encontrar una gran cantidad de información y documentación, el problema al realizar una búsqueda es que la información se encuentra ordenada en listas según el tema y no tiene un sistema de búsqueda, por lo que hay que tener claro el tema que se desee buscar.

Desde la página principal de la wiki podemos acceder a los módulos, donde hay una lista de proyectos con prioridades (P1, P2), cada uno de estos proyectos tiene su propia

página web. Descubrimos que NetBeans dispone de un sistema IRC de mensajería instantánea.

En la lista de correos nbdiscuss@netbeans.org podemos encontrar que se utiliza un sistema de votaciones para decidir realizar un plugin y para la elección de la nueva junta directiva. En nbdev@netbeans.org la mayor parte de mensajes plantean dudas, se dan respuestas con explicaciones escribiendo directamente código o proporcionando enlaces, especialmente de documentación en la wiki. Una observación, los propios usuarios en ocasiones encuentran la solución a sus problemas y lo hacen pública de modo que otros usuarios puedan resolver el mismo problema con más facilidad. En dev@platform.netbeans.org podemos encontrar alguna petición pero sin respuesta o pendientes para realizar, por la carga de trabajo que supone o su poca relevancia.

En el foro de desarrolladores no encontramos muchas propuestas y en el foro de usuarios encontramos algunas pero sin respuesta. El foro y las listas de correos están conectadas, pero la cuenta del foro de NetBeans es distinta a la utilizada en los correos. Y en general no se observa mucha participación, puede ser porque la comunidad no es muy numerosa o en NetBeans puede que no haya ninguna persona comprometida a revisar los foros o correos, en contraste con OpenOffice donde había varios desarrolladores que respondían a la mayoría de las dudas.

Miércoles 25 de junio de 2014

La siguiente comunidad sobre la que centro mi investigación es Eclipse. En su página web se encuentra documentado el proceso de desarrollo que aplican, de modo que busqué si se explicaba también en el resto de comunidades, pero no tuve mucho éxito en lo que se refiere a procesos de requisitos. Encontré el proceso que siguen en NetBeans para el tratamiento de *issues* o problemas, que al parecer es el habitual entre los usuarios de Bugzilla, porque OpenOffice y Eclipse también siguen dicho proceso.

Me resultó complicado encontrar una lista de correos con el tema adecuado, puesto que las listas son muy específicas para proyectos concretos y no parecen recomendadas para dudas de usuarios. Se puede observar que en algunas listas de correos no hay mucha actividad, como en platform-dev. Descubrimos en eclipse.org-committers un mensaje para organizar una reunión de contribuidores por Hangout, de lo cual no hay documentación. También encontramos mensajes sobre el proyecto para rediseñar la interfaz de la página oficial de Eclipse, donde se creó una entrada en Bugzilla para su

gestión y en un enlace a un blog se observa que se construyó un prototipo, que ya no está disponible, y una encuesta para conocer la opinión de los usuarios.

Jueves 26 de junio de 2014

Continué con la observación y el análisis de las listas de correo y foros de Eclipse. En el foro hay una sección expresamente para propuestas. Hay una propuesta en el foro con una encuesta de votos, pero al parecer ya existe un proyecto en marcha y no hay muchos votos. Las propuestas de nuevos proyectos pasan a [Ec Propuestas]. Estas propuestas provienen principalmente de desarrolladores que ya están contribuyendo al proyecto o piensan colaborar en ellos.

En la página de propuestas se puede ver en qué fase se encuentra el proyecto e información sobre el mismo, como la descripción o los participantes e interesados. Cada propuesta tiene asignado un proyecto padre donde se gestiona el proyecto, las descargas, quienes están involucrados, la documentación y los recursos de los desarrolladores. Por cada proyecto se suele crear una lista de correos, lo que nos dificultaba identificar en qué lista queríamos buscar o contribuir.

Viernes 27 de junio de 2014

Por último nos dedicamos al estudio de la comunidad Mozilla, que tras el primer acceso a su página oficial y por las dificultades que encontré para navegar por su página web lo dejé en último lugar. Debido a la gran cantidad de proyectos que tiene para restringir el estudio nos centramos en su proyecto Firefox, puesto que es su proyecto principal. Accedí a la página de la comunidad hispana pero no había mucha actividad.

El foro con formato de listas de correo, en un principio, me resultó un tanto confuso, pero después de analizar las otras comunidades me resultó más fácil comprender su funcionamiento. Los mensajes de los usuarios se archivan en GoogleGroups, no es necesario registrarse en Firefox para publicar mensajes así que se puede encontrar spam en las listas de correos.

En la lista de mozilla.support.firefox la mayor parte de mensajes son errores o dudas de usuarios resueltas por otros usuarios o desarrolladores, se ofrecen soluciones con enlaces que redireccionan a la sección de Extensiones de Firefox. Por parte de los desarrolladores en firefox-dev encontramos muchas discusiones. En un email encontramos una petición para añadir una explicación a la documentación sobre la discusión de revisar el desarrollo iterativo, que no es un problema relevante a resolver y

puesto que no tiene mucho apoyo se suspende por ahora la idea. Pero de todos modos podemos observar que queda documentado en la wiki. En otras propuestas, parece ser de desarrolladores, se discute su necesidad y si son aceptadas pasan a Bugzilla para su seguimiento y desarrollo. Parece ser que la persona que acepta la propuesta es quién se hace responsable de desarrollarla. Una observación que podemos apreciar es que en la sección de desarrolladores no se tarda tanto en contestar a los mensajes, son más activos, mientras que la sección de usuarios no es tan activa.

Martes 1 de julio de 2014

He de realizar una aportación a las comunidades, de modo que tras pensar en varias opciones decido realizar una encuesta para completar mi estudio. Puesto que no se me ocurrían buenas propuestas de nuevas funcionalidades para el software y ya había observado cómo reaccionaban las comunidades ante nuevos requisitos. Las preguntas de la encuesta se dirigen a los objetivos del estudio, diferenciando entre usuarios y desarrolladores.

Miércoles 2 de julio de 2014

Utilizando los formularios que ofrece GoogleDrive realizo la encuesta en inglés y en español. Tras registrarme en las distintas comunidades publico las encuestas en los foros y espero la respuesta y la colaboración de los participantes.

Jueves 3 de julio de 2014

No recibimos mucha participación en los foros de modo que decidimos enviar las encuestas directamente a los usuarios de aplicaciones OSS, obteniendo así la opinión de los usuarios sobre el proceso de requisitos, lo que completará los casos de estudio realizados sobre las comunidades OSS más populares. Luego me dedico a completar mi análisis de cada comunidad respondiendo a las preguntas planteadas al principio.

Viernes 11 de julio de 2014

Tras una iteración sobre lo realizado decidimos realizar aportaciones sencillas a otras comunidades y realizar una entrevista a un contribuidor de una comunidad más pequeña, Social Network Manager, para conseguir una aproximación desde el punto de vista de un desarrollador.

Anexo B. Ejemplos de Mensajes

A continuación, se muestran varios ejemplos de mensajes que podemos encontrar en las listas de correos o foros de las comunidades. Estos mensajes nos muestran datos que consideramos relevantes en esta investigación como la toma de decisiones mediante votos o discusiones, ejemplos de propuestas o peticiones de requisitos y las reacciones obtenidas.

Ejemplo 1: La duda encontrada en la lista de correos openoffice-users pasa a ser considerada como una posible mejora que se añade a Bugzilla para su seguimiento y desarrollo.

Asunto: no icon for Track Changes Fecha: Jueves, 01 Mayo 2014
Autor: Ivan (Usuario) Hello, I do NOT find any icon for your "Track Changes" feature; please advise how I create or customize such an icon. Actually, it could be very helpful to have it on the toolbar. Thank you so much for your time! Kind regards Ivan Penev
Autor: Dave (Desarrollador) Information about creating custom toolbar icons can be found in chapter 17 of the Writer User Guide: https://wiki.openoffice.org/wiki/Documentation/OOo3_User_Guides/Chapters The guide itself is a little out of date, but the toolbar customization instructions are still valid for AOO version 4.x.x Dave
Autor: Ivan (Usuario) But, actually I do not find there any icon for the Track Changes feature to drag and drop it to the toolbars.... Can you please assist me step by step with that issue? Thank you! Kind regards Ivan Penev
Autor: Dave (Desarrollador) I do not know if there is a specific icon for "Track Changes" included in the standard icon set. A quick look through the set suggests to me that there is not one for this specific function. However, you can use any icon from the standard icon set, or import your own. It does not matter if the icon originally represented another function, the only important thing is that you know what it represents on your toolbar. Dave
Autor: Dave (Desarrollador) Rereading your reply, I do not understand what you mean by "drag and drop", because I do not know how that can be done. Please read chapter 17 of the Writer User Guide at the link I gave you before, which explains in detail how to do what you want. Dave

Autor: Brian (Desarrollador)
o Open a text (Writer) document.
o Either: go to Tools | Customize... | Toolbars; under OpenOffice.org
Toolbars, select your preferred toolbar.
o Or: click the down arrow at the extreme right of your preferred toolbar and select Customize toolbar... .
o Click Add... .
o Under Category, select Edit.
o Under Commands, scroll down and select Record.
o Click Add and Close.
o Your new icon appears in the list of Commands; use the up and down arrows to move it to our preferred position in the toolbar.
o OK.

The new Record button acts as a toggle, of course.
Sorry: no drag and drop.

I trust this helps.

Brian Barker

Autor: Robert (Desarrollador)
On my Web page at http://audilab.bme.mcgill.ca/~funnell/swil_ooo.html are two icons that I created for this purpose, and a pointer to instructions for associating icons with commands. The instructions given above by Brian Barker can then be used to add the commands and their new icons to the toolbar.

- Robert

Autor: Alan (Desarrollador)
Ivan, please be sure to review the chapter Dave refers to. The link to the specific chapter is https://wiki.openoffice.org/w/images/8/86/0217WG3-CustomizingWriter.pdf As regards drag and drop to tool bar, it is likely that is a reference to the customization process.

Select "View | Toolbars | Customize..." from the menu.
The "Customize" dialog opens prepared for editing the Standard toolbar. If the "Add..." button is clicked an "Add Commands" dialog appears with two list boxes, "Category" and "Commands". Entries in the "Commands" list box can be dragged into the "Toolbar Content Commands" list box of the "Customize" dialog.

The drag and drop doesn't happen directly onto the toolbar in Writer. It is done between the "Add Commands" and "Customize" dialog boxes.

The commands for revision marking are found in two categories in the "Add Commands" dialog.

In the "Add Commands" dialog select "View" from the "Category" list and then find "Show" on the "Commands" list. Add that to the desired toolbar by drag and drop or by using the "Add" button.

The other revision marking commands will be found by selecting "Edit" from the "Category" list. The relevant commands that are now available in the "Commands" list are "Accept Change", "Accept or Reject", "Protect Records", "Record" and finally "Reject Change".

None of these commands has a default icon so you will need to select your own icon for each so the button isn't just a flat gray box on the modified toolbar.

Autor: circulars (Desarrollador)
The thread has prompted me to put up a couple of long-overdue enhancement requests:

124791 - drag & drop icons
https://issues.apache.org/ooo/show_bug.cgi?id=124791

124794 - a toolbar icon for every command.
https://issues.apache.org/ooo/show_bug.cgi?id=124794

'Record' is not very obvious. It might be worth considering putting up one suggesting changing the 'icon' to 'track changes' or better, Robert Funnell's icon.

En la Figura 2 encontramos la mejora registrada en Bugzilla, donde se puede observar en más detalle la discusión sobre el tema.

Figura 2. Ejemplo 1 de una mejora para OpenOffice registrada en Bugzilla

Ejemplo 2: En la lista de correos openoffice-users volvemos a encontrar una petición de un requisito, donde se produce una larga discusión en la que participan usuarios y desarrolladores. A continuación recogemos varios mensajes de esta discusión.

Asunto: Suggestion.
Fecha: Martes, 13 Mayo 2014

Autor: Sarala (Usuario)
Sir / Madam,

For many years I used WordPerfect as my Word Processor and Desktop Publisher to produce a 12 page newsletter. I have never found a better program. As I now have iMac I use Openoffice and have found that satisfies most of my requirements.

However there is one very important property (if that's the right word) that WP had that is missing from all the Word Processors that I have used. That is what WP called "Reveal Codes", where every change that was made in the document was shown by a particular code. e.g.: Hard return was "HRT". If this was not what was wanted then you could make the change

you required. Or as sometimes happens in OpenOffice, something happens which I don't understand, I have no way of finding out why.
Would it be possible to incorporate this feature in OpenOffice? It would enhance this program's appeal to me and, I'm sure to many others who still hanker for the user friendliness of WP.
Gordon Lee.

Autor: Jim McLaughlin (Usuario)
I would certainly "second" Gordon Lee's suggestion for ver. 5.x.x that having a parallel to the Corel/WP "reveal Codes" function would be invaluable.

Autor: Doug (Desarrollador)
This subject came up a couple of months ago, maybe with LO instead of OO, and the stuffy answers came back and said you have to use "styles," which is a giant ball of wax. In other words, even if you could see what was happening, you could not change it "on the fly," you had to go and figure out what kind of style would make the text look like what you wanted and and create that style. And then your whole document would have that style. What a humongous mess! WordPerfect Office 12 word processor (only*) will run in 32-bit Linux, but I don't know whether the reveal styles function works--not everything does. Also it apparently will not run in 64-bit systems, but maybe that can be finagled.

*Do not try to install other features of the office suite, like the spread sheet, or Corel Draw, etc.

--doug

Autor: Andrea (Desarrollador)
This has been discussed at length for years. OpenOffice historically has always used styles instead. If some developers want to work on such a feature, their contribution is surely welcome (and they can get help on our dev list), but at the moment there are no "official" efforts towards this. See the long discussion at https://issues.apache.org/ooo/show_bug.cgi?id=3395 for the current status.

Gordon: you received many answers. If you didn't see them, you are not subscribed, see the archives:
http://openoffice.apache.org/mailing-lists.html#users-mailing-list-public

Regards,
Andrea.

Autor: Bruce Byfield (Desarrollador)
By the way, the idea of a Reveal Codes feature goes way back - over a decade, in fact:
https://issues.apache.org/ooo/show_bug.cgi?id=3395
And a discussion of the issues involved with implementing it:
https://wiki.openoffice.org/wiki/Reveal_Codes_-_Alternative_for_OOo_Writer
Other discussion at LibreOffice:
http://comments.gmane.org/gmane.comp.documentfoundation.libreoffice.bugs/87075

Autor: Marina (Usuario)
Sorry I can't help Julian, I have found no tutorials at all. I'm used to learning by reading the manual and then lots of trial and error, and after investing many hours doing just that, I have found that using styles can save some time with complex documents.

(I write and translate books, so using styles was "forced" on me by my editors, a dozen years ago or so. As of today, I am still sort of unsure what amount of time I have *effectively* saved by learning how to use styles - but I was given no option, and now
that I've grown accustomed to styles, it's possible I am starting to save time. Twelve years, and many books down the track....! >:])

To those who chimed in to justify styles: it is quite obvious to me that you are missing the point. For starters, it sounds like you don't really know WordPerfect, and imagine "Reveal codes" to be something other than it was.

But more basically, OpenOffice is for people like Julian and me. If people like me and Julian put forward a suggestion, it should be the programmer's job to consider it from the user's perspective... or shouldn't it??

Again, I would like to say thank you to those who give their time for free to build and improve OpenOffice. This includes those who put forward useful suggestions "from the user's perspective" :-)

marina

Más adelante otro usuario apoya la moción y pregunta si se puede votar, a lo que le responden que ya existe una discusión sobre el tema y lo redireccionan a Bugzilla.

Asunto: Reveal Codes Query
Fecha: Viernes, 16 Mayo 2014

Autor: japples (Usuario)
Does a list exist for OO users to vote for proposed future functionalities? I would like to vote for Reveal Codes.

Reveal Codes is a very useful tool. It works independently of the "Style rules" and does not require the user to take time to search "style rules" then create or edit (causes changes in other documents?)
Logically, it seems the more "style rules" the less efficient OO Writer.

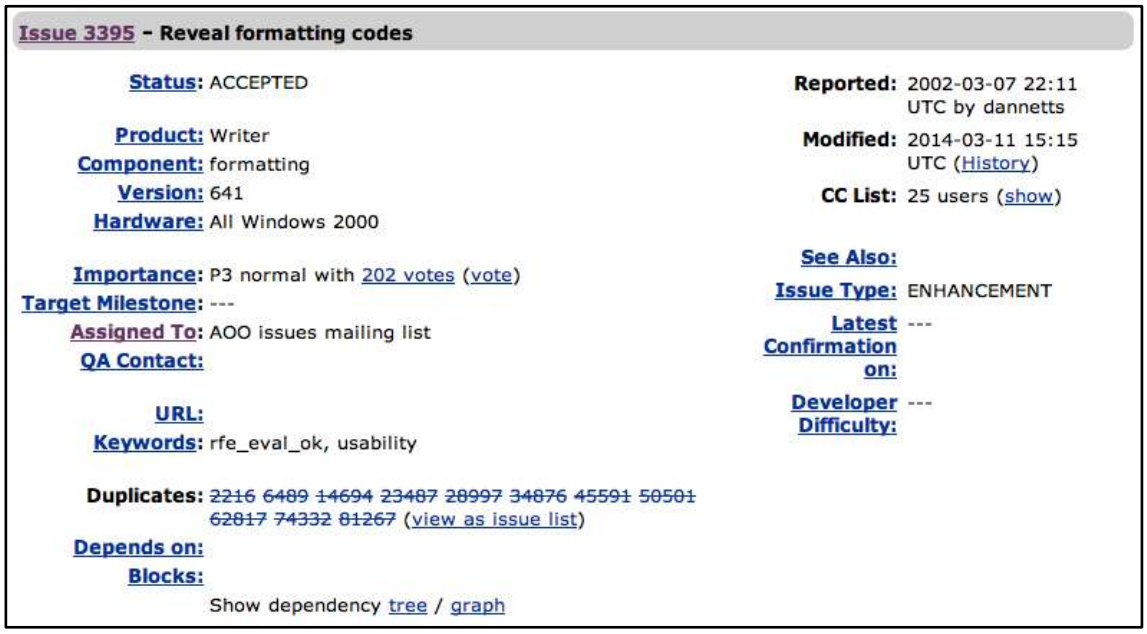
Guess you could say, reveal codes is like lifting the hood of a vehicle to change spark plugs. While "style rules" feature is similar to re-configuring the vehicle to operate bypassing spark plugs (takes more time and effort by the user).

Thanks for your patience -
Jack

Autor: Andrea (Desarrollador)
The link I've already sent https://issues.apache.org/ooo/show_bug.cgi?id=3395 is all we have, but it has this functionality. You can vote, you can comment (but no need to do so unless you have something important to add to the discussion) and you can see the existing discussion and work in progress.

Regards,
Andrea.

En la Figura 3 encontramos la mejora registrada en Bugzilla.



Issue 3395 - Reveal formatting codes

Status:	ACCEPTED	**Reported:**	2002-03-07 22:11 UTC by dannetts
Product:	Writer	**Modified:**	2014-03-11 15:15 UTC (History)
Component:	formatting	**CC List:**	25 users (show)
Version:	641		
Hardware:	All Windows 2000	**See Also:**	
Importance:	P3 normal with 202 votes (vote)	**Issue Type:**	ENHANCEMENT
Target Milestone:	---	**Latest Confirmation on:**	---
Assigned To:	AOO issues mailing list	**Developer Difficulty:**	---
QA Contact:			
URL:			
Keywords:	rfe_eval_ok, usability		

Duplicates: ~~2216~~ ~~6489~~ ~~14694~~ ~~23487~~ ~~28997~~ ~~34876~~ ~~45591~~ ~~50501~~ ~~62817~~ ~~74332~~ ~~81267~~ (view as issue list)

Depends on:

Blocks:

Show dependency tree / graph

Figura 3. Ejemplo 2 de una mejora para OpenOffice registrada en Bugzilla

Ejemplo 3: Una propuesta en el foro español que ya se ha solicitado con anterioridad y se permite votar (Figura 4).

Figura 4. Ejemplo 3 de una mejora para OpenOffice en el foro

Ejemplo 4: Se realiza un *brainstorming* utilizando la plataforma de Google Moderator.

Asunto: Help us brainstorm ideas for Apache OpenOffice 4.0
Fecha: 21 Agosto 2012 Autor: Rob Weir (Desarrollador) As we perform the final preparations to release Apache OpenOffice 3.4.1 it is a good time to look ahead to the future. A big opportunity is OpenOffice 4.0. That once seemed so very far away, but 2013 is getting closer every day. Will it be a large collection of small ideas? Will it have a major overarching theme? Or will it just be whatever random stuff we happen to have on a given date when we release 4.0? The answer, of course, depends on what we, as project members/volunteers decide to do. It is a good time now, as a background activity, to poll the community and wider ecosystem on ideas for Apache OpenOffice 4.0. To participate, go to this page on Google Moderator, where you can help us gather and rate ideas: https://www.google.com/moderator/#16/e=2011d5 A few project members have already "seeded" this with some initial ideas. Of course, you are encouraged to add your own ideas, as well as rate the ideas of others. Try not to censor yourself from thinking outside-of-the-box. We need big ideas as well as incremental ones. We don't have a close date on this brainstorming activity, but it is good to get your ideas in early, so there is an opportunity for others to rate and comment on it. Regards, -Rob
Fecha: 24 Agosto 2012 Autor: Reizinger Zoltán (Desarrollador) In old days around 2008-2009 we made similar brainstorming on OOo Base features, and we put all ideas into wiki page: http://wiki.openoffice.org/wiki/Base/Features/Pool It is a "wild ideas list" not prioritized most of them valid for present state of Base. Regards, Zoltan
Fecha: 25 Septiembre 2012 Autor: Rob Weir (Desarrollador) Today, a week later: 633 users - 527 ideas - 7,607 votes So we are still getting a good amount of feedback. I added a mention of this brainstorming on the www.openoffice.org website header. That should give this even more visibility. I've heard from some that it would be good to get to a point where we can take a "snapshot" of the feedback received, and process that, to help set priorities for AOO 4.0. Would it make sense to do that in another week, say on October 1st? At that point we can: 1) Put a "thank you" note on the Google Moderator page and stop accepting new suggestions. Point the users to the ooo-users or ooo-dev mailing list instead, 2) Export the ideas and scores received so far to a CSV file and archive that someplace. 3) Discuss the results received 4) Maybe a blog post to highlight the brainstorming activity and the results received? Any other ideas?

I don't need to own any of this, but since I started it I'm willing to finish it. But if anyone else wants to take a lead on this, please volunteer.

Regards,

-Rob

Autor: Shengeng Liu (Desarrollador)
Hi, all,
I went through top 180 ideas, and here is my summary till now:

1. OOXML has 12 ideas, and totally 989 + votes
2. Modern UI has 11 ideas, and totally 347 + votes
3. Loading performance has 4 ideas and got 310 + votes
4. MS binary format support has 4 ideas and 269 + votes
5. Toolbar/sidebar related ideas are 8, and 257 + votes
6. 2 ideas ask for patch/update, and got 253 + votes
7. Preference setting related ideas are 8 and 244 + votes
8. Copy/paste related ideas are 4 and 207 + votes
9. Install improvement related ideas are 4 and got 176 + votes
10. Mobile support ideas are 4 and got 171 + votes

About 40 ideas are requiring new features, and got >1300 + votes
17 ideas are to improve user experience for various operations, and got 400 + votes
9 ideas are about social integration, and got 383 votes
13 asking for the enhancement of existing functions, and got 377 + votes

It is difficult to categorize the ideas and count. I wonder if Kevin can define some criteria and we can pick up different areas to build up the report.

- Simon

Ejemplo 5: Encontramos una llamada para votar la publicación de la versión 4.1.0 de Apache OpenOffice.

Asunto: [VOTE] Release Apache OpenOffice 4.1.0 (RC4)

Fecha: 25 Abril 2014
Autor: Jürgen Schmidt (Desarrollador)
Hi all,

this is a call for vote on releasing the available release candidate (RC4) as Apache OpenOffice 4.1.0.

Apache OpenOffice 4.1 is a minor update with many bugfixes and at least 2 major improvements. It's the first version where we have the iAccessibility2 support integrated and available. A very huge step forward to reach and better support disabled users especially on Windows. The second improvement is the switch to 64 bit on MacOS. A long and overdue "must do" shift forward to support newer APIs (replace deprecated APIs) and platforms on MacOS. And we can provide again more complete UI translations and have now support for 38 languages. New languages for this release compared to 4.0.1 are Bulgarian, Danish, Hebrew, Hindi, Norwegian Bokmal and Thai.

Apache OpenOffice 4.1 will be a further key milestone to continue the success of OpenOffice.

An overview of the integrated release issues can be found under:
http://people.apache.org/~jsc/milestones/4.1.0-rc4/AOO4.1.0_RC4_fixes.html

The RC4 fixed 2 further problems:
https://issues.apache.org/ooo/show_bug.cgi?id=124682
https://issues.apache.org/ooo/show_bug.cgi?id=124701

RC3:
https://issues.apache.org/ooo/show_bug.cgi?id=124617
https://issues.apache.org/ooo/show_bug.cgi?id=124639

RC2:
https://issues.apache.org/ooo/show_bug.cgi?id=124599
https://issues.apache.org/ooo/show_bug.cgi?id=124607
https://issues.apache.org/ooo/show_bug.cgi?id=124509
https://issues.apache.org/ooo/show_bug.cgi?id=124394

The release candidate artifacts (source release, as well as binary releases for 38 languages) and further information how to verify and review Apache OpenOffice 4.1.0 can be found on the following wiki page:
https://cwiki.apache.org/confluence/display/OOOUSERS/Development+Snapshot+Builds

(alternative directly via http://ci.apache.org/projects/openoffice/milestones/4.1.0-rc4)

*.dmg files are currently not recognized as binaries and have to be saved manually (save link as ...).
The RC is based on the release branch AOO410, revision 1589052! And a fresh and clean RAT scan output of this revision can be found under

http://people.apache.org/~jsc/milestones/4.1.0-rc4/AOO4.1.0_RAT_Scan.html

Please vote on releasing this package as Apache OpenOffice 4.1.0

The vote starts now and will be open until:
Monday, 28 April: 2014-04-28 10:00pm UTC+2.

But we invite all people to vote (non binding) on this RC. We would like to provide a release that is supported by the majority of our project members.

[ ] +1 Release this package as Apache OpenOffice 4.1.0
[ ] 0 Don't care
[ ] -1 Do not release this package because...

Fecha: 25 Abril 2014
Autor: imacat (Desarrollador)
+1 from imacat

Fecha: 25 Abril 2014
Autor: Jürgen Schmidt (Desarrollador)
The vote period to release the AOO 4.1.0 RC4 (based on release branch AOO410, revision 1589052) as Apache OpenOffice 4.1 has ended.

The ballot results in 28 votes, 28 +1 votes including 12 binding PMC member votes.

3 binding +1 votes are necessary for the release. That means the ballot closed successful to release the RC as AOO 4.1 Beta.

Vote tally

+1 imacat (binding)
+1 Николай Нинков
+1 Markus Lange (binding)
+1 Kay Schenk (binding)
+1 V Stuart Foote
+1 Armin Le Grand (binding)
+1 Yueshen Fan
+1 Steve Yin
+1 Edwin Sharp
+1 Regina Henschel (binding)
+1 Shengfeng Liu
+1 Pedro Albuquerque
+1 Dirk Groskamp
+1 Josef Latt
+1 Nityanand Sharma
+1 Kazunari Hirano
+1 Aivaras Stepukonis
+1 Mathias Röllig
+1 Don Harbison (binding)
+1 Andrea Pescetti (binding)
+1 Juergen Schmidt (binding)
+1 Oliver Rainer Wittmann (binding)
+1 Herbert Duerr (binding)
+1 Andre Fischer (binding)
+1 Jeongkyu Kim
+1 Pedro Giffuni
+1 Andrew Rist (binding)
+1 Tal Daniel

Ejemplo 6: En firefox-dev por parte de los desarrolladores encontramos muchas discusiones. En un email encontramos una petición para revisar el desarrollo iterativo, lo que después de una larga discusión se decide que no es un problema relevante a resolver y puesto que no tiene mucho apoyo se suspende por ahora la idea. Pero queda documentado en la wiki.

Asunto: Proposed Revision to Iterative Development
Fecha: 13 Mayo 2014
Autor: Anthony Hughes

Hello all,

I was supposed to talk about this today in the Iteration Planning meetings but I was having Vidyo issues which I could not sort out in time. As an alternative I'm emailing you all the following proposal. I'd like to request that we make a couple of minor tweaks to the current Iterative Development process.

The first revision is in regards reducing the traffic around [qa?] bugs. Over the first few iterations we've noticed a couple different classifications of [qa?] bugs which could be immediately tagged [qa-] by you, without QA needing to step in. These bugs include work breakdown bugs, UX/polish bugs, and test-only bugs. QA would like to request that when selecting work, you immediately flag these bugs as [qa-]. As we scale up this will reduce some of the churn and make our lives a bit easier.

The second revision is to adopt a policy around near-merge changes, and keeping this policy in mind when selecting work for the final iteration. We would like to ask that work selected in the final days of the final iteration be looked through the same lens as if you were asking for uplift. In other words, if the work would be considered medium to high-risk then we'd ask that the patch is held off until after migration occurs; especially for changes landing a day or two before merge. The main reason for this is that testing will not be able to be completed until soon after the migration. If we find issues the risk to backing out changes on Aurora or Beta could be high in certain circumstances. We think this policy will reduce churn and reduce the likelihood of messy back-outs, thereby making all our lives a bit easier.

To be clear, neither of these revisions is coming as a result of something that has gone wrong. We merely see an opportunity to prevent something from happening in the future. I've documented the proposal here:
https://wiki.mozilla.org/QA/Desktop_Firefox/Walkthroughs/Iteration_Development#v2_Proposal

Do these revisions seem reasonable?

Thank you,

Anthony Hughes
Senior Test Engineer
Mozilla Corporation

RESPUESTAS

- Proposed Revision to Iterative Development *Anthony Hughes*
  - Proposed Revision to Iterative Development *Jared Wein*
    - Proposed Revision to Iterative Development *Gervase Markham*
      - Proposed Revision to Iterative Development *Gijs Kruitbosch*
      - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Richard Newman*
      - Proposed Revision to Iterative Development *Lawrence Mandel*
      - Proposed Revision to Iterative Development *Matthew N.*
    - Proposed Revision to Iterative Development *Anthony Ricaud*
  - Proposed Revision to Iterative Development *Nick Alexander*
  - Proposed Revision to Iterative Development *Benjamin Smedberg*
    - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Benjamin Smedberg*
      - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Gavin Sharp*
      - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Gavin Sharp*
      - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Gijs Kruitbosch*
      - Proposed Revision to Iterative Development *Gavin Sharp*
      - Proposed Revision to Iterative Development *Anthony Hughes*
      - Proposed Revision to Iterative Development *Lawrence Mandel*
      - Proposed Revision to Iterative Development *Anthony Hughes*
    - Proposed Revision to Iterative Development *Mike Connor*
    - Proposed Revision to Iterative Development *Gavin Sharp*
    - Proposed Revision to Iterative Development *Irving Reid*
      - Proposed Revision to Iterative Development *Anthony Hughes*

Ejemplo 7: Entre varias sugerencias encontramos otra propuesta en Mozilla en la que se discute su necesidad y finalmente se acepta y pasa a gestionarse en Bugzilla.

Asunto: proposal for changing Nightly/Aurora's default profile behavior
Fecha: 21 Mayo 2014
Autor: Gavin Sharp

The problem: testing Aurora/Nightly builds side-by-side with your "normal" Firefox requires some manual profile management, which is a barrier for web developers and other testers who want to just download-and-run.

This problem has been discussed here before.
https://bugzilla.mozilla.org/show_bug.cgi?id=895030 covers one proposal, and previous threads here include:

https://mail.mozilla.org/pipermail/firefox-dev/2013-July/000574.html (
https://groups.google.com/forum/#!topic/firefox-dev/P6lyKEaSRbM/discussion)
https://mail.mozilla.org/pipermail/firefox-dev/2013-July/000560.html (
https://groups.google.com/forum/#!topic/firefox-dev/23sGBTc_rM0/discussion)

Here's a new proposal that I've come up with, which is a modification of a revised proposal that Jeff Griffiths and the devtools folks came up with:

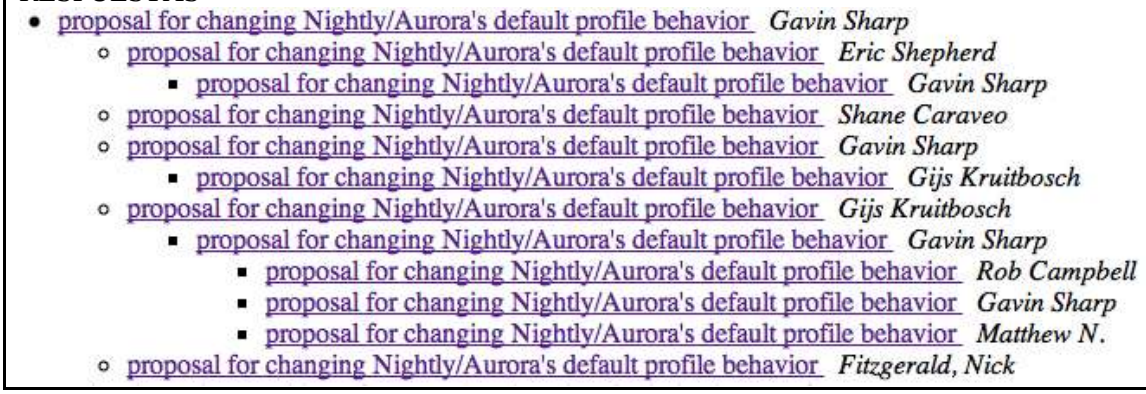
https://wiki.mozilla.org/User:GavinSharp/Profile-proposal

One of the benefits of this proposal is that it does not affect existing Nightly/Aurora users.

I'd like broader feedback on this proposal before we move to implement.

Gavin

RESPUESTAS



En la Figura 5 encontramos la mejora registrada en Bugzilla.

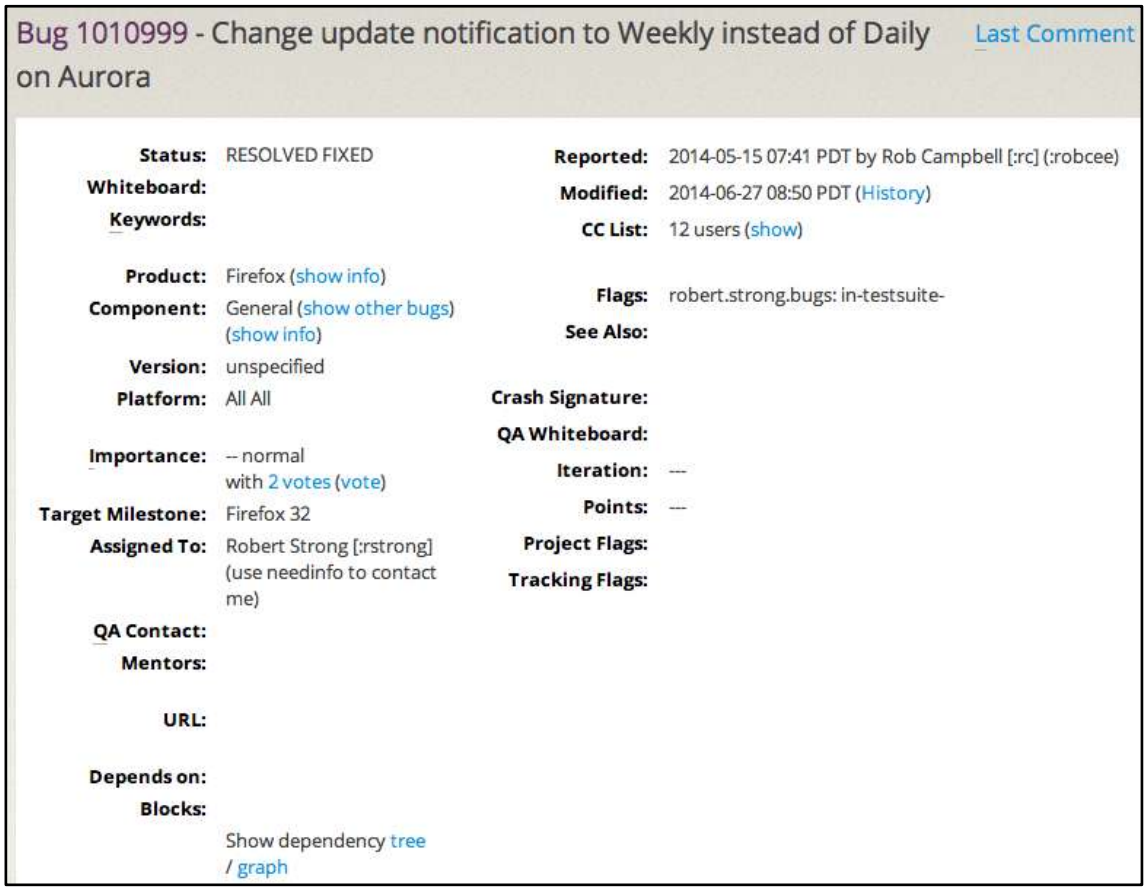


Figura 5. Ejemplo 7 de una mejora para Mozilla registrada en Bugzilla

Ejemplo 8: En la lista de correos nbdiscuss@netbeans.org de NetBeans podemos encontrar el sistema de votaciones para la realización de un plugin y para la elección de la nueva junta directiva.

Asunto: [nbdiscuss] Vote for the 21st NetBeans Governance Board! Fecha: 04 Febrero 2014 Autor: Tinuola Awopetu
Hello NetBeans Community, The Governance Board election <https://netbeans.org/community/news/show/1608.html> is underway, and here are the final nominees who need your votes: * Tim Boudreau * Lou Dasaro * Glenn Holmer * Benno Markiewicz * Zoran Sevarac Learn more about the nominees <https://netbeans.org/community/articles/election_profiles.html> and their involvement with

NetBeans, and then cast your votes <https://netbeans.org/community/articles/nbelections.html>. (/Note: To see and use the voting form, you MUST be logged in to your netbeans.org account./)

The voting period runs through midnight, *Monday, February 17th* in the last time zone. The results and the new board will be announced *Tuesday, February 18th*.

Good luck to the nominees!

Asunto: [nbdiscuss] The 21st NetBeans Governance Board: Tim Boudreau and Zoran Sevarac!
Fecha: 19 Febrero 2014
Autor: Tinuola Awopetu

The NetBeans team is pleased to announce the new community-elected members of the 21st cycle of the Governance Board: *Tim Boudreau and Zoran Sevarac*.

The Oracle-appointed third member of the Board will be *Ashwin Rao*, NetBeans Senior Product Management Group Manager.

Thank you to all the great nominees <https://netbeans.org/community/articles/election_profiles.html> for taking part in this election and for their continued support of the NetBeans project.

Thank you to the NetBeans community for voting!

For more information:
NetBeans Governance <https://netbeans.org/about/os/governance.html>
Board Election Process <https://netbeans.org/about/os/election-process.html>
Current and Past Board Members <https://netbeans.org/about/os/who-board.html>

Ejemplo 9: En la lista de desarrolladores de NetBeans encontramos varias propuestas que quedan pendientes de realizar debido a que no son relevantes.

Asunto: [platform-dev] Adding a button next to a minimize button

Fecha: 15 Junio 2014
Autor: Nummern
Hi all,

I have a table in a TopComponent and would like to add some extra options to filter out rows an d sort ordering in the table. I'd like to have these options next to the minimize button.

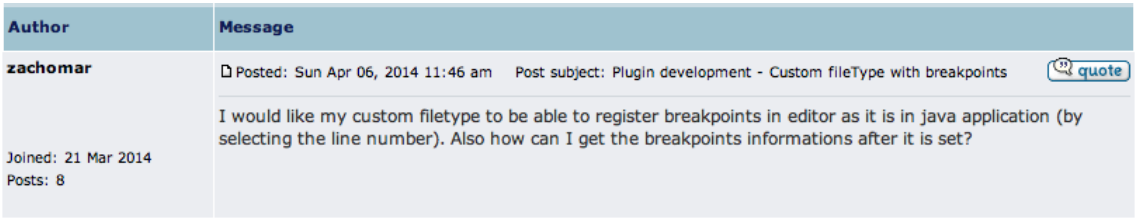
The reason for this is that the plugin I am building is in the style of an Eclipse plugin which offe rs the same functionality. Is there a way to do this? This is not a major issue if it is not possible but it would be nice to have.

Thanks,
Martin

Fecha: 17 Junio 2014
Autor: Tim Boudreau
There is no support for contributing components to the tabs or the space
next to them. If you were really determined, you'd have to write your own
UI delegate for the tabs themselves and hack it into UIManager early in
startup - not trivial and not recommended.
-Tim

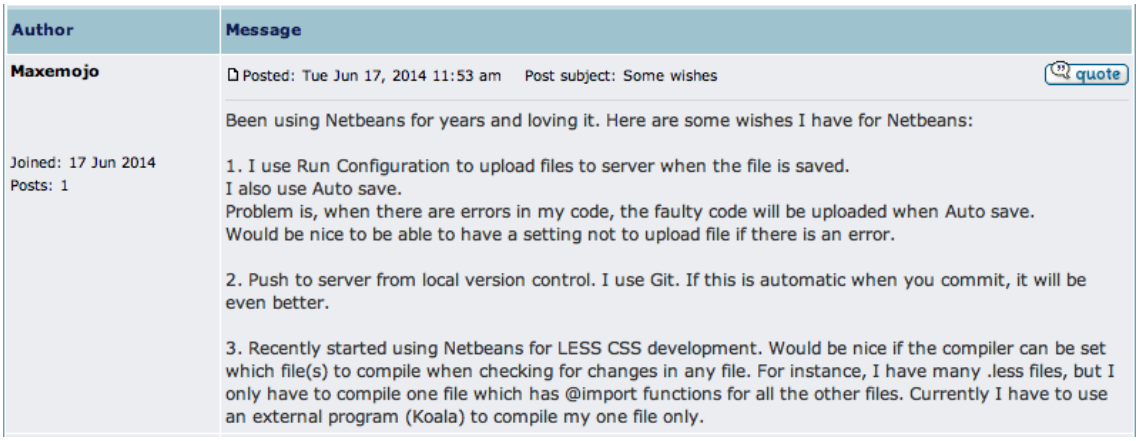
Fecha: 20 Junio 2014 Autor: mone.java Do you found a soultion? I'm also interested!
Fecha: 24 Junio 2014 Autor: Nummern not yet, it's not too high on my requirements list at the moment but I will update when I get to it and if I find a solution or alternative. Cheers, Martin

Ejemplo 10: En los foros tanto de desarrolladores como de usuarios de NetBeans encontramos peticiones pero sin respuestas (Figuras 6 y 7).



Author	Message
zachomar Joined: 21 Mar 2014 Posts: 8	Posted: Sun Apr 06, 2014 11:46 am Post subject: Plugin development - Custom fileType with breakpoints quote I would like my custom filetype to be able to register breakpoints in editor as it is in java application (by selecting the line number). Also how can I get the breakpoints informations after it is set?

Figura 6. Ejemplo 10 de una mejora para NetBeans en el foro de desarrolladores



Author	Message
Maxemojo Joined: 17 Jun 2014 Posts: 1	Posted: Tue Jun 17, 2014 11:53 am Post subject: Some wishes quote Been using Netbeans for years and loving it. Here are some wishes I have for Netbeans: 1. I use Run Configuration to upload files to server when the file is saved. I also use Auto save. Problem is, when there are errors in my code, the faulty code will be uploaded when Auto save. Would be nice to be able to have a setting not to upload file if there is an error. 2. Push to server from local version control. I use Git. If this is automatic when you commit, it will be even better. 3. Recently started using Netbeans for LESS CSS development. Would be nice if the compiler can be set which file(s) to compile when checking for changes in any file. For instance, I have many .less files, but I only have to compile one file which has @import functions for all the other files. Currently I have to use an external program (Koala) to compile my one file only.

Figura 7. Ejemplo 10 de una mejora para NetBeans en el foro de usuarios

Ejemplo 11: Eclipse tiene una sección en el foro expresamente para propuestas. Las propuestas de nuevos proyectos pasan a [Ec Propuestas]. Estas propuestas provienen principalmente de desarrolladores que ya están contribuyendo al proyecto o piensan colaborar. En la Figura 8 mostramos el mensaje en el foro donde se puede observar el enlace a su ficha informativa.

Figura 8. Ejemplo 11 de una propuesta para Eclipse en el foro

Ejemplo 12: En los foros de Eclipse hay una propuesta con una encuesta, pero ya existe un proyecto en marcha y no hay muchos votos (Figura 9).

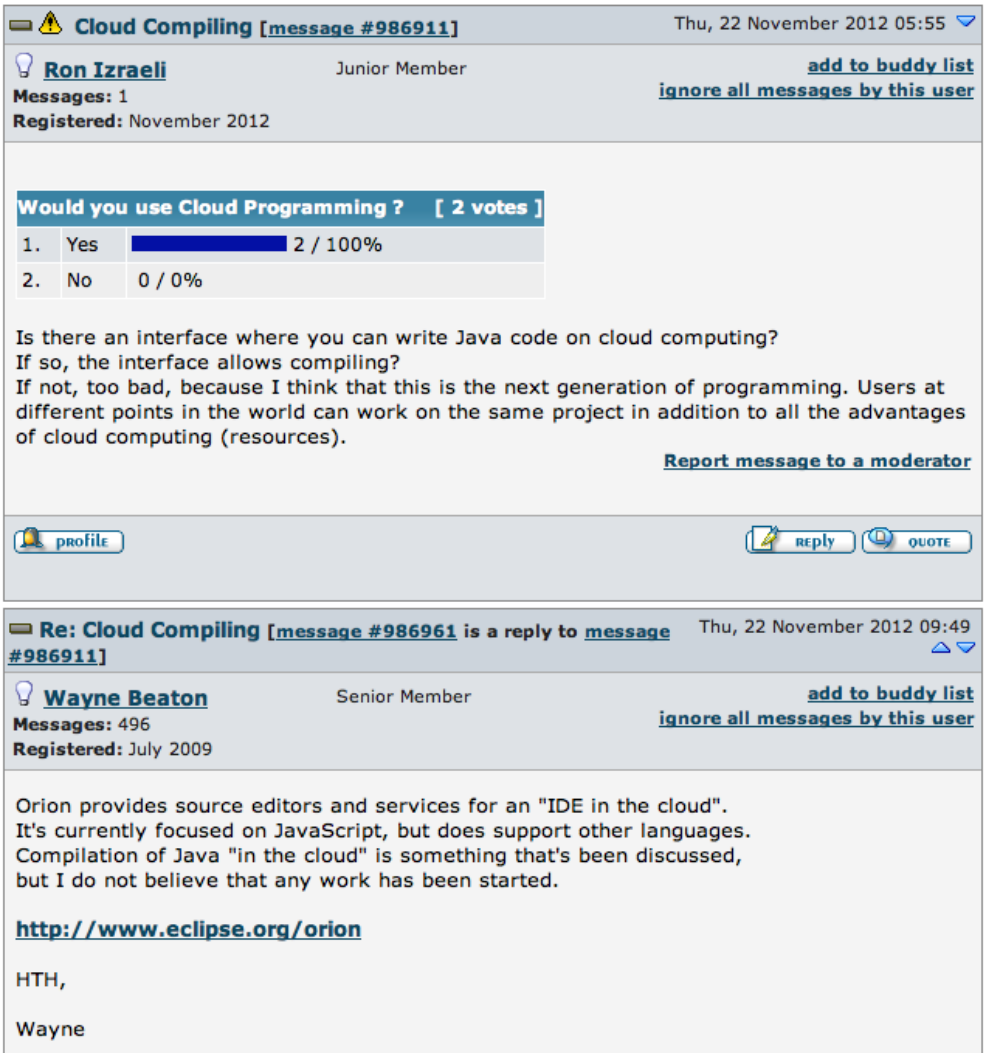


Figura 9. Ejemplo 12 de una mejora para Eclipse en el foro

Ejemplo 13: En eclipse.org-committers podemos encontrar una reunión de contribuidores por Hangout, de lo cual no hay más documentación.

Asunto: Eclipse Committers Hangout starts at 11am EDT Fecha: 30 Mayo 2014 Autor: Wayne Beaton

Click here to join:

https://plus.google.com/hangouts/_/g3w5lf65tr3zg4eevv3yrtddwma

Wayne
--
Wayne Beaton
Director of Open Source Projects, The Eclipse Foundation
Learn about Eclipse Projects

Ejemplo 14: Encontramos el proyecto dedicado a rediseñar la interfaz de la página web de Eclipse donde se creó un bug en Bugzilla y un blog donde se aprecia que se construyó un prototipo (Figura 10), que ya no está disponible, y una encuesta para conocer la opinión de los usuarios.

Asunto: Eclipse.org new look and feel

Fecha: 06 Mayo 2014
Autor: Christopher Guindon
The Eclipse Foundation is currently working on a new look and feel for www.eclipse.org. We have setup a test instance on http://staging.eclipse.org. We are planing to go live on June 11th, 2014.

Project Status

We want to make it as easy as possible for the project web sites to be migrated to the new theme, called Solstice. We've created an assessment report (below) for each project to identify the work required to migrate. Please take a look at this report to determine how your project web site will function with Solstice.

https://docs.google.com/spreadsheet/ccc?key=0AsQLOtRdqwM9dHBEcGdUYVU3WmFTQXpySGNUNG9xX3c&usp=drive_web#gid=0

Updating Project Sites

We would like to have as many of the project pages migrated to Solstice by June 11. You can begin to fix your site using the Solstice theme on the staging.eclipse.org server. To use the staging server, please create a branch called "staging" in your project's current web site git repository. The staging server will publish content from that staging branch instead of master, but only if it exists. If you commit a change to the staging branch, please be patient. It'll take 5 to 10 minutes before you can see an update on the staging server.

If you have any questions regarding testing, please contact me, chrisguindon, on #eclipse-fdn, #eclipse and #eclipse-dev on freenode.net (IRC) or by email chris.guindon@xxxxxxxxxxx.

If you plan on opening a bug, please mark it as a blocker to bug 432342.

Regards
Christopher

Related links
1. Eclipse.org Website Redesign 2014
http://wiki.eclipse.org/Eclipse.org_Website_Redesign_2014

2. Bug 432342 - Eclipse.org Website Redesign 2014
https://bugs.eclipse.org/bugs/show_bug.cgi?id=432342

3. Time for a new eclipse.org L&F
http://ianskerrett.wordpress.com/2014/03/31/time-for-a-new-eclipse-org-lf/

--
Christopher Guindon
Senior Web Developer
Eclipse Foundation
Twitter: @chrisguindon

Fecha: 26 Mayo 2014
Autor: Henrik
Hi,

frequently projects use on their project main page a collection of four navigation buttons for Documentation, Download, Getting Involved and Support.
These buttons come with images which should IMHO be replaced for the new look and feel.

Does the foundation offer those replacements?

Thanks,
Henrik

Fecha: 28 Mayo 2014
Autor: Christopher Guindon
Hi Henrik,

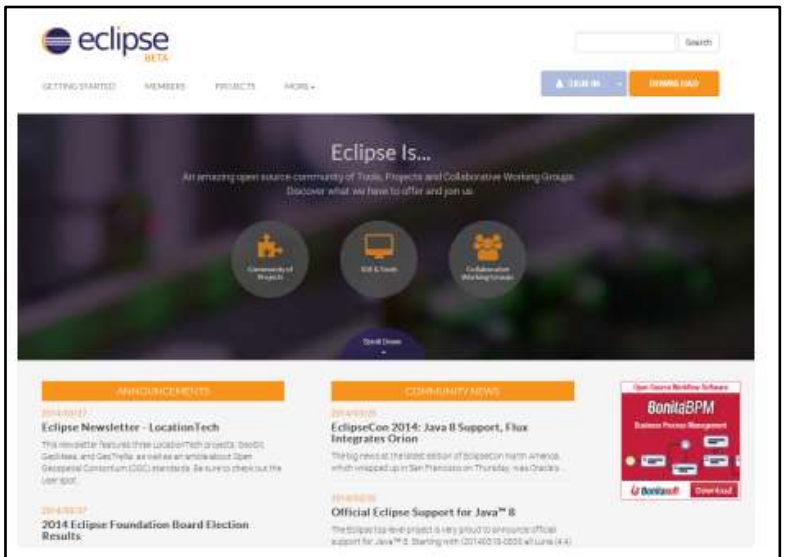
The four button navigation was not created by us but I added some custom CSS to Solstice to support it.

For example, the egit project website is using it:
http://staging.eclipse.org/egit

I created Bug 436108 - Update navigation buttons for Documentation, Download, Getting Involved and Support for project pages

Unfortunately, I don't expect this to be ready for June 11th, 2014.

Cheers,
--
Christopher Guindon
Senior Web Developer
Eclipse Foundation
Twitter: @chrisguindon



When we started we set out some goals for the new design:

1. Make the site look nicer, ie. reduce some of the clutter, be more consistent with colours, etc.
2. The web site should be responsive so it is usable on mobile devices.
3. Make it easier for newcomers to get started at Eclipse.
4. Help people discover and understand our Eclipse Working Group

To achieve these goals we put ogether a team of people to help design and advise on the new site. Chris Aniszczyk and Sven Efftinge provided feedback and advice from the committer/project perspective. Nitin Dahyabhai and Russ Bateman helped provide the perspective of the newcomer to the Eclipse community. Finally, Christopher Guindon, Denis Roy, myself and Matt Joanisse, a graphic designer hired by the Foundation to work on the site, worked on the design, content and implementation. I'd like to thank all of these people for spending the time helping get us to this stage.

As I mentioned, now we would like to get feedback on the prototype from the community. In particular, we would like to know the following:

1. Overall, what is your view of the site? We have started a survey to solicit your feedback.
2. Have we missed some key content that should be referenced from the navigation menu or from the key landing pages?
3. Is the Getting Started Page missing any important content you think newcomers would find useful?

Please provide the feedback via this survey. As with any creative initiative, we guarantee we will read all feedback but we might not agree to implement it.

Figura 10. Ejemplo 14 de la nueva interfaz de Eclipse

Anexo C. Encuesta

En la Figura 11 se muestran las preguntas realizadas en la encuesta a los usuarios de aplicaciones OSS.

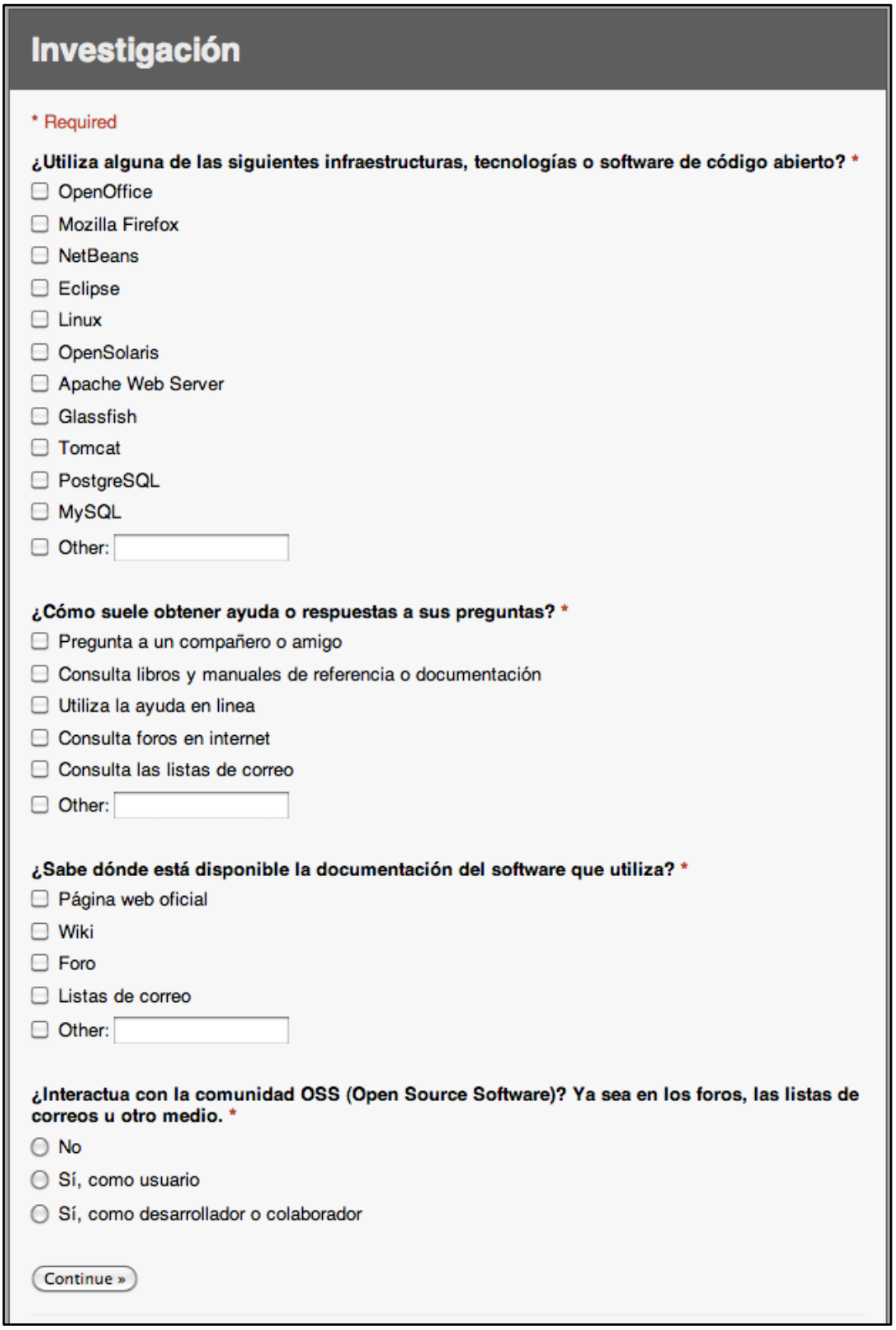
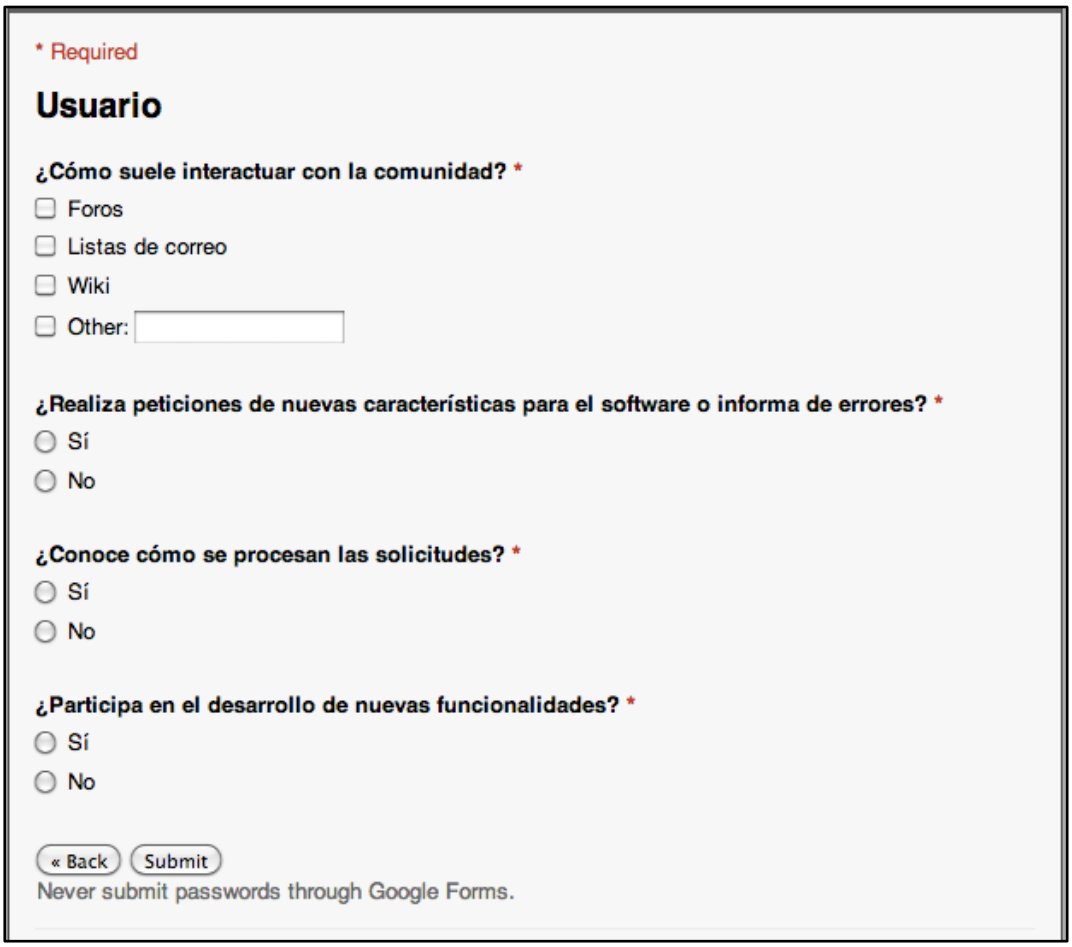


Figura 11. Encuesta a usuarios de OSS



* Required

Usuario

¿Cómo suele interactuar con la comunidad? *

- [ ] Foros
- [ ] Listas de correo
- [ ] Wiki
- [ ] Other:

¿Realiza peticiones de nuevas características para el software o informa de errores? *

- ( ) Sí
- ( ) No

¿Conoce cómo se procesan las solicitudes? *

- ( ) Sí
- ( ) No

¿Participa en el desarrollo de nuevas funcionalidades? *

- ( ) Sí
- ( ) No

« Back Submit

Never submit passwords through Google Forms.

Figura11. Encuesta a usuarios de OSS (Continuación)

Anexo D. Herramientas

A continuación se describen las herramientas que se han utilizado durante la colaboración en este proyecto de investigación empírica.

Mantis

Mantis Bug Tracker, www.mantisbt.org, es un software que al igual que Bugzilla sirve para gestionar tareas en un equipo de trabajo, destaca por su facilidad y flexibilidad de instalar y configurar. Esta aplicación se utiliza para probar soluciones, hacer un registro histórico de alteraciones y gestionar equipos de forma remota.

Esta aplicación fue empleada para proponer una nueva funcionalidad, que consistía en implementar un botón que permitiese volver al explorador de archivos, manteniendo el directorio en el que se estaba navegando. La Figura 12 muestra un extracto de la pantalla de creación de un *Bug*.

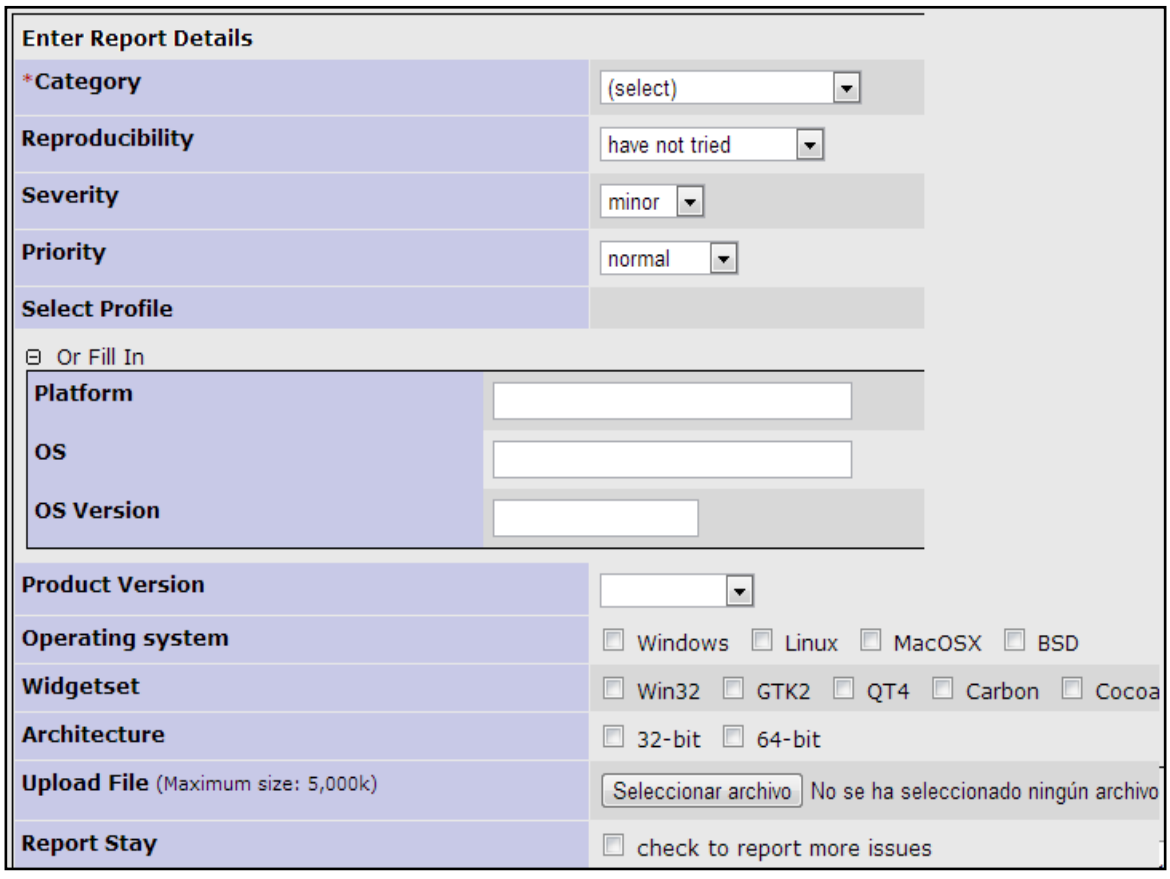


Figura 12. Creación de un reporte en Mantis

Una vez se ha realizado una aportación, ésta pasa por los siguientes estados:

- **New**: Nueva propuesta/*bug* reportado.
- **Feedback**: Encontramos esto en errores reportados que ya han sido solucionados en versiones siguientes del producto.
- **Acknowledged**: Significa que el *bug* ha sido admitido como tal pero no tienen pensado arreglarlo en el presente.
- **Confirmed**: Confirmado el error a falta de asignarlo.
- **Assigned**: Error asignado.
- **Resolved**: El error/propuesta reportado ya ha sido resuelto.
- **Closed**: El *bug* ha sido cerrado por invalidez.

FileZilla's Trac

La página de FileZilla cuenta con un sistema para la creación de Tickets [Fz Trac], que sirve para reportar errores, proponer nuevas funcionalidades y ver los parches subidos, dividido en los diferentes módulos que forman la aplicación (Figura 13).

	Open	Pending further information	Closed	Search
FileZilla Client:	Bug reports (471) Feature requests (736) Patches (25)	Bug reports (31) Feature requests (1) Patches (7)	Bug reports (2425) Feature requests (1001) Patches (86)	Summary: Description: Search
FileZilla Server:	Bug reports (72) Feature requests (138) Patches (0)	Bug reports (4) Feature requests (0) Patches (1)	Bug reports (232) Feature requests (164) Patches (16)	Summary: Description: Search
Other:	Bug reports (7) Feature requests (8) Patches (1)	Bug reports (1) Feature requests (0) Patches (0)	Bug reports (1285) Feature requests (344) Patches (46)	Summary: Description: Search

Figura 13. Sistema de gestión de Filezilla

Al navegar por la herramienta se pueden buscar tickets por tema, ordenar por fecha, añadir comentarios a tickets existentes y crear nuevos. La Figura 14 muestra el formato de la creación de un ticket.

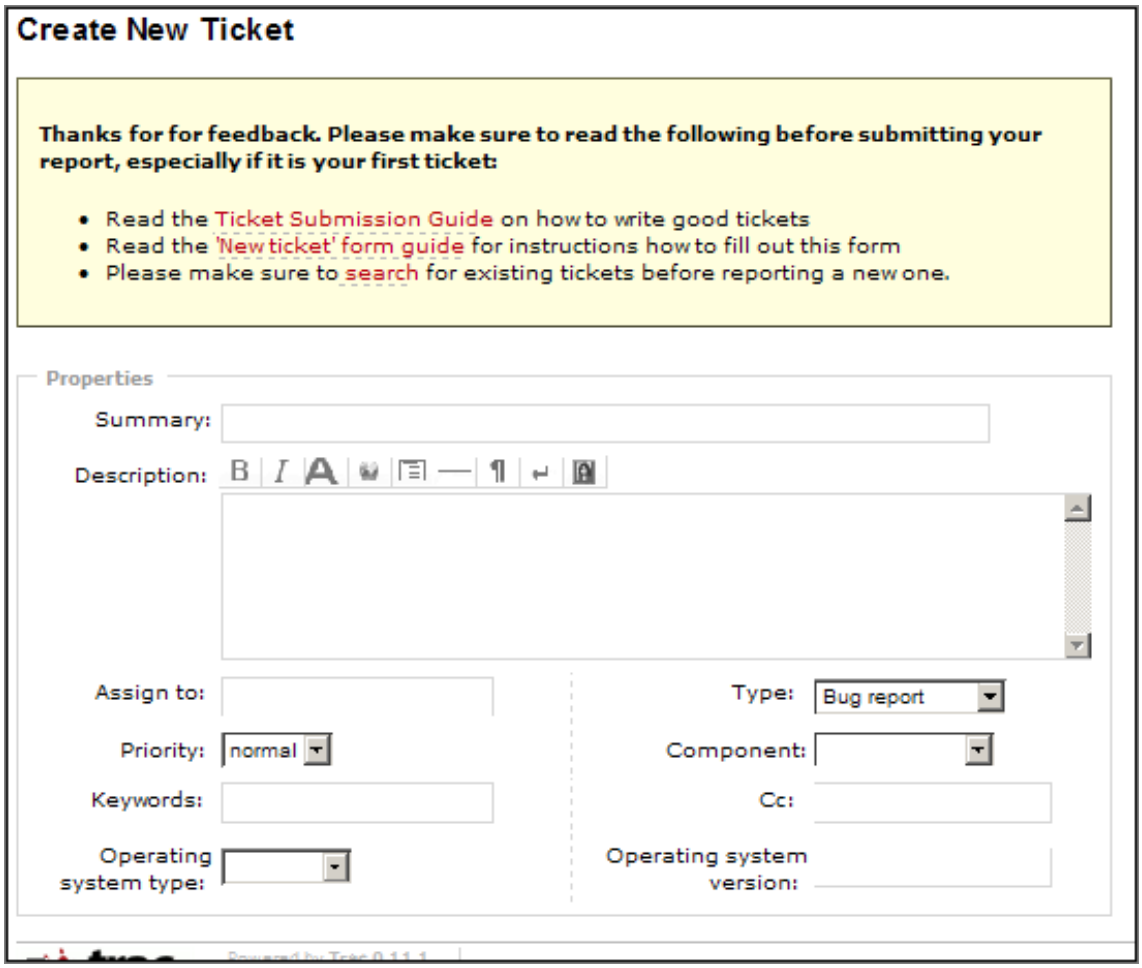


Figura 14. Creación de un reporte en FileZilla's Trac

Una vez se ha realizado una aportación, ésta pasa por los siguientes estados:

- **New**: Nueva propuesta/*bug* reportado que eventualmente llegará al estado closed.
- **Closed**: El error/propuesta reportado ya ha sido resuelto.
- **Assigned**: Error asignado.
- **Accepted**: Significa que el *bug* ha sido admitido.
- **Reopened**: Encontramos esto en errores o propuestas reportados que ya habían sido cerrados y vuelven a ser un tema de interés.
- **Moreinfo**: Informa que el ticket enviado necesita más información.